Manual del Usuario

xperTools © 2007

Montevideo – Uruguay

Índice

EL DISEÑADOR (DESIGNER) 4

LA BARRA DE MENÚ 4
LA VENTANA DE CONTROLES 6
BASIC CONTROLS 7
ADVANCED CONTROLS 8
COMPOUND CONTROLS 9
CONTROL CONTAINERS 10
ART &SHAPES 11
LA VENTANA DE PROPIEDADES 12
LA VENTANA DE PROPIEDADES DE ESTILO 13
LA VENTANA DE DATOS 14
LA VENTANA DE EVENTOS 14
EL OBJETO CONTROLLER Y OTROS PARÁMETROS EN EVENTOS 16
JERARQUÍA DE INTERFACES 22
IXPERTCONTROL 22
IXPERTVALUECONTROL 24
IXPERTDATACONTROL 25
IXPERTCONTAINER 26
IXPERTMULTICONTAINER 27

EL EDITOR DE HOJAS DE ESTILO (STYLER) 28

ANEXO CLASES DE CONTROLES 29

XPERTLABEL: IXPERTVALUECONTROL 29
XPERTLINK: IXPERTVALUECONTROL 29
XPERTBUTTON: IXPERTCONTROL 29
XPERTCHECK: IXPERTVALUECONTROL 29
XPERTTEXT: IXPERTVALUECONTROL 30
XPERTNUMBER: IXPERTVALUECONTROL 30
XPERTDATE: IXPERTVALUECONTROL 30
XPERTCOMBO: IXPERTVALUECONTROL, IXPERTDATACONTROL 30
XPERTPROGRESS: IXPERTCONTROL 30
XPERTSLIDERV: IXPERTDATACONTROL 31
XPERTSLIDERH: IXPERTDATACONTROL 31
XPERTIMAGE: IXPERTVALUECONTROL 31
XPERTCLIP: IXPERTVALUECONTROL 31
XPERTRADIO: IXPERTVALUECONTROL, IXPERTDATACONTROL 32
XPERTLIST: IXPERTDATACONTROL 32
XPERTTREE: IXPERTDATACONTROL 32
XPERTGRID: IXPERTDATACONTROL 33
XPERTCHARTC: IXPERTDATACONTROL 33

xpertChartL: IxpertDataControl 33
xpertChartP: IxpertDataControl 34
xpertMenuBar: IxpertDataControl 34
xpertShelf: IxpertDataControl 34
xpertMap: IxpertDataControl 34
xpertTextRtf: IxpertValueControl 34
xpertTextArea: IxpertValueControl 34
xpertMask: IxpertValueControl 34
xpertColor: IxpertValueControl 34
xpertSelector: IxpertDataControl 35
xpertUploader: IxpertControl 35
xpertAccordionH: IxpertMultiContainer, IxpertDataControl 35
xpertAccordionV: IxpertMultiContainer, IxpertDataControl 35
xpertDividerH: IxpertMultiContainer, IxpertDataControl 35
xpertDividerV: IxpertMultiContainer, IxpertDataControl 35
xpertTab: IxpertMultiContainer, IxpertDataControl 36
xpertBook: IxpertMultiContainer, IxpertDataControl 36
xpertBox: IxpertContainer 36
xpertPanel: IxpertContainer 36
xpertLine ← xpertBox 36
xpertTriangle ← xpertBox 36
xpertSquare ← xpertBox 36
xpertPentagon ← xpertBox 36
xpertHexagon ← xpertBox 36
xpertCircle ← xpertBox 36
xpertArrow ← xpertBox 36

ANEXO SINTAXIS DE EVENTOS 37

Estructuras de Control 37
Operadores 37
Constantes Globales 40
Funciones Globales 40

ANEXO INSTALACIÓN Y CONFIGURACIÓN 42

Introducción: 42
Prerrequisitos: 42
Edición WEB 42
Edición Desktop 42
Estructura de Directorios: 42
Componentes de las RIATools: 43
Configuración de RIATools: 44
Configuración de RIATools sobre IIS: 44
Configuración de RIATools sobre Tomcat: 47
Troubleshooting: 49

El Diseñador (Designer)

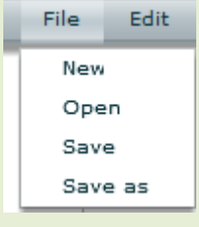
El diseñador (o designer de las Riatools) es un editor WYSIWYG de frontends. En él puede diseñar y programar la interfaz de usuario, controlar la estructura, aspecto y funcionalidad de la capa de presentación de sus aplicaciones (web o desktop) y controlar la interacción con sus aplicaciones de backend vía scripts asociados a eventos.

Algunas de las características más relevantes que ofrece son:

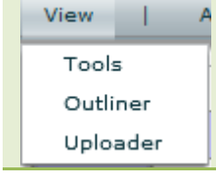
- Rico set de controles de alto impacto visual y usabilidad para el programador y para el usuario final.
- Permite realizar a distancia el desarrollo y el soporte de las aplicaciones realizadas.
- Describe los frontends 100% como metadatos.
- Ofrece una Alta Portabilidad entre plataformas (Windows, Linux, Mac)
- Ofrece una Alta portabilidad en el cliente tanto en browsers como Desktop (IE, Firefox, Safari, Eudora, AIR App, .NET App, Java App)
- Se integran con variedad de backends (Genexus 8, 9 y 10, .NET, J2EE, PHP/MYSql, PL-SQL, T-SQL) donde puede implementar las reglas de negocio y acceso a datos.

La Barra de Menú

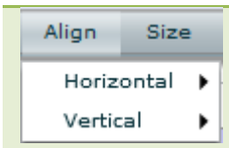
Situada en la parte superior del diseñador, la barra de menú le permite realizar las siguientes tareas:



New	Crea un nuevo formulario
Open	Abre el dialogo de selección de formularios para abrir uno previamente salvado en el repositorio
Save	Salva el formulario actual (al final de la barra de menú se muestra el nombre del formulario en edición). Si es un formulario nuevo (el nombre de formulario esta en blanco) se abre el dialogo de salvado de formularios para indicar la ubicación y nombre en el repositorio.
Save As	Abre el dialogo de salvado de formularios para indicar la ubicación y nombre en el repositorio con que se salvara una copia del formulario actual.

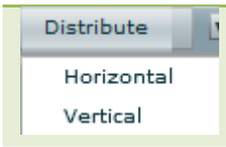


Tools	Oculta/Visualiza la ventana de controles
Outliner	Oculta/Visualiza la ventana de eventos
Uploader	Oculta/Visualiza la ventana de upload



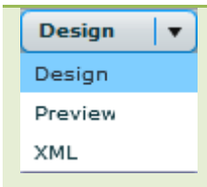
Left	Alinea a la izquierda un grupo de controles seleccionados
Center	Alinea a la centro horizontal un grupo de controles seleccionados
Richt	Alinea a la derecha un grupo de controles seleccionados
Top	Alinea hacia arriba un grupo de controles seleccionados
Middle	Alinea a la al centro vertical un grupo de controles seleccionados
Bottom	Alinea a la hacia abajo un grupo de controles seleccionados

Width	Iguala el ancho de los controles seleccionados
Height	Iguala el alto de los controles seleccionados
Size	Iguala ancho y alto de los controles seleccionados

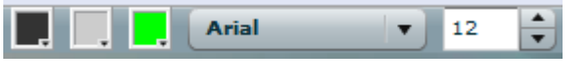


Horizontal	Distribuye la distancia horizontal entre los controles seleccionados
Vertical	Distribuye la distancia vertical entre los controles seleccionados

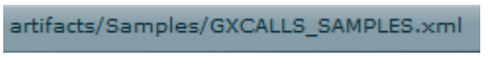
NOTA: Para todas las acciones que aplican sobre el conjunto de controles seleccionados, la guía es el último control seleccionado.



Design	Pasa a modo Design, en el cual se puede editar el formulario (agregar, quitar, mover, y cambiar propiedades y eventos de los controles)
Preview	Pasa a modo Preview, en el cual se puede interactuar con el formulario como lo va a hacer el usuario final
XML	Pasa a modo XML en el que se visualizan los METADATOS XML que describen el formulario actual

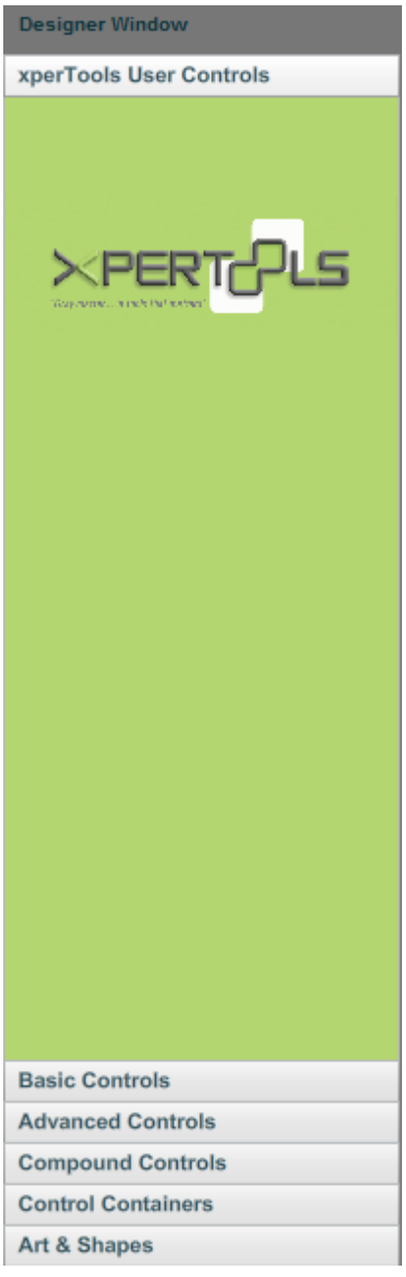


Permite seleccionar el color de fondo, letra, tema, tipo y tamaño de letra por defecto.



Muestra la ruta del repositorio y el nombre del formulario que se está editando.

La Ventana de Controles



Situada por defecto (es una ventana flotante) sobre el margen derecho de la pantalla, agrupa por familia -para simplificar la ubicación- más de 40 controles que Ud. dispone para diseñar la GUI de sus formularios. Los agrupamientos actuales (esta previsto incluir nuevos en futuras versiones) son los siguientes:

- Basic Controls
- Advanced Controls
- Compound Controls
- Control Containers
- Art & Shapes

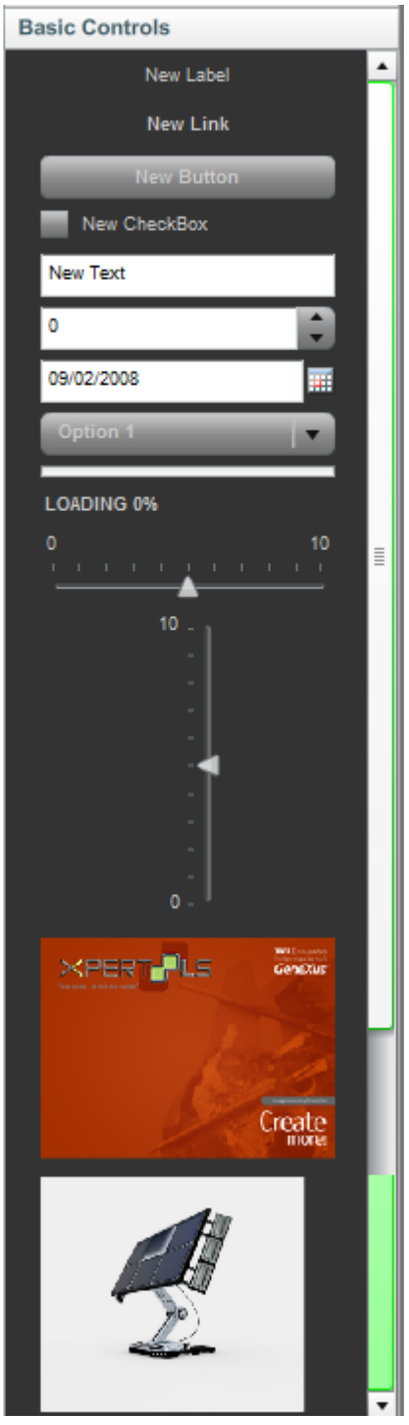
Para agregar un control en su formulario Ud. Debe seleccionar el grupo de controles, ubicar el control que desea agregar (se muestra tal cual cómo funcionan dentro de la toolbox), seleccionarlo, arrastrarlo al formulario y soltarlo próximo al lugar donde desearía ubicarlo (ya sea en el contenedor principal -main- o en algún contenedor previamente agregado), luego puede seleccionarlo y moverlo o cambiarle el tamaño directamente en el formulario usando los grab handlers.

Todos los controles tienen propiedades, propiedades de estilo y métodos y eventos. Puede ver un detalle de los mismos, para cada tipo de control, en el Anexo Clases de Controles, estas propiedades se presentan y pueden ser modificadas en la Ventana de Propiedades, también puede ver y modificar el aspecto mediante la Ventana de Propiedades de Estilo, los juegos de datos en la Ventana de Datos y los scripts asociados a los eventos en la Ventana de Eventos.

Además, los controles están agrupados en distintas interfaces por funcionalidad común, puede ver las propiedades comunes en la sección Jerarquías de Interfaces

Con los controles de la toolbox podrá confeccionar interfaces de usuario antes nunca alcanzables con otras herramientas y podrá hacerlo de manera fácil y rápida. Además, gracias a la representación de las GUI generadas basándose en metadata, podrá migrar de arquitectura, plataforma, framework de desarrollo (Genexus, .NET, Java, AJAX), sin ningún esfuerzo. Aproveche toda la lógica de negocios de sus aplicaciones (Genexus, .NET, J2EE, Store Procedures PL-SQL o T-SQL) dotándolas de una excelente capa de presentación en unas pocas horas de trabajo.

Basic Controls



Dentro de los controles básicos (en el grupo correspondiente de la toolbox) puede encontrar los siguientes controles:

- Etiquetas (xpertLabel) [*]
- Vínculos (xpertLink) [*]
- Botones (xpertButton)
- Casillas de Verificación (xpertCheck) [*]
- Cuadros de Texto (xpertText) [*]
- Cuadros Numéricos (xpertNumber) [*]
- Cuadros de Fechas (xpertDate) [*]
- Listas desplegables (xpertCombo) [*+]
- Barras de Progreso (xpertProgress)
- Barra deslizante (xpertSliderH y xpertSliderV) [*]
- Imágenes (xpertImage) [*]
- Animaciones y Clips de Video (xpertClip) [*]

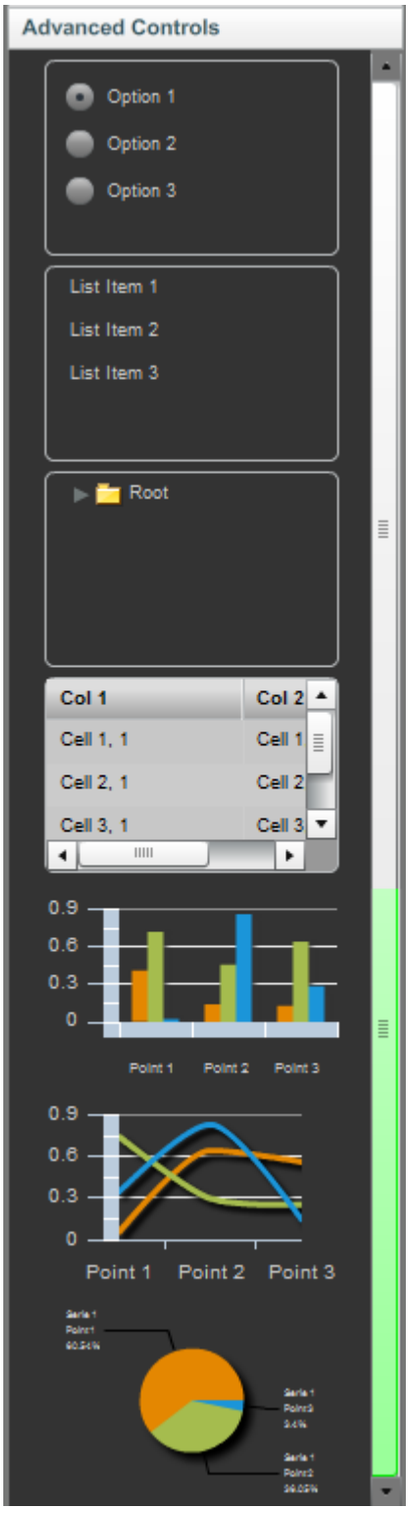
Todos estos controles comparten las propiedades, métodos y eventos de la interfaz IxpertControl, los marcados con [*] implementan además la interfaz IxperValueControl y los marcados con [+] la interfaz IxpertDataControl

Para ver el detalle de propiedades, métodos y eventos de cada uno de estos controles vea la sección Anexo Clases de Controles

Con los controles básicos puede resolver los elementos básicos característicos de cualquier formulario de ingreso y visualización de datos.

Simples de usar, e integrar, con las propiedades suficientes para customizarlos, y los eventos necesarios para manipularlos programáticamente o integrarlos al manejo de datos de sus aplicaciones de backend, los controles básicos le permiten lograr lo mismo que hoy dispone en su plataforma de desarrollo pero de forma mucho más rápida, sencilla y potente y con un look and feel de altísima calidad gracias al esquema de gráficos vectoriales combinado con el manejo de hojas de estilo dinamicas que le permiten definir temas o skins con sus propiedades de visualización seleccionables por el usuario final.

Advanced Controls



Dentro de los controles avanzados (en el grupo correspondiente de la toolbox) puede encontrar los siguientes controles:

- Grupos de opciones (xpertRadio) [*+]
- Listas (xpertList) [+]
- Arboles (xpertTree) [+]
- Grillas (xpertGrid) [+]
- Tile Repeaters (xpertTile) [+]
- Graficas de Columnas (xpertChartC) [+]
- Graficas de Lineas (xpertChartL) [+]
- Graficas de Torta (xpertChartP) [+]

Todos estos controles implementan las siguientes interfaces:

- IxpertControl
- IxpertDataControl

Los grupos de opciones (Radio Button) implementan además la interfaz IxpertValueControl

Para ver el detalle de propiedades, métodos y eventos de cada uno de estos controles vea la sección Anexo Clases de Controles

Con los controles avanzados puede presentar y manipular grupos de datos, por los que son aptos para hacer data binding. Todos ellos implementan servicios para cargarse desde resultados generados por web services.

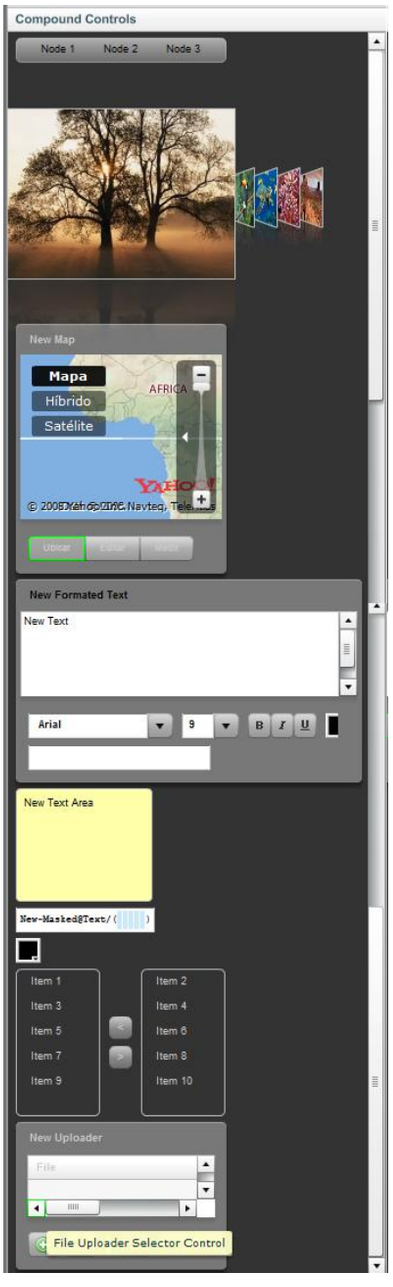
Las capacidades graficas de este set de controles, sumadas a la funcionalidad que prestan harán extremadamente fácil el armado de la capa de presentación de sus aplicaciones, incluso para el manejo de datos estructurados y complejos y de grades volúmenes.

Los usuarios de sus aplicaciones valoraran la calidad de presentación de información así como la facilidad de uso de la GUI.

Incluya un manejo de graficas en su aplicación tan fácil y rápida como arrastrar y soltar el control adecuado de la toolbox.

La performance que presentan estas herramientas no dejaran de sorpréndelo, a Ud. Y a sus usuarios. Maneje formularios de ingreso de datos con cientos o miles de controles sin notar degradaciones de performance en sus aplicaciones. Presente grandes volúmenes de datos en pantalla, con o sin manejo de paginado a velocidades increíbles.

Compound Controls



Dentro de los controles compuestos o combinados (en el grupo correspondiente de la toolbox) puede encontrar los siguientes controles:

- Menues (xperMenu) [+]
- Galerías de Imágenes (xpertShelf) [+]
- Mapas (xpertMaps) [+]
- Editor de Texto con formato (xpertTextRtf) [*]
- Aéreas de Texto con Formato (xpertArea) [*]
- Cuadros de Texto enmascarados (xpertMask) [*]
- Selector de Colores (xpertColor) [*]
- Selector de Datos (xpertSelector) [+]
- Uploader de archivos (xpertUploader)

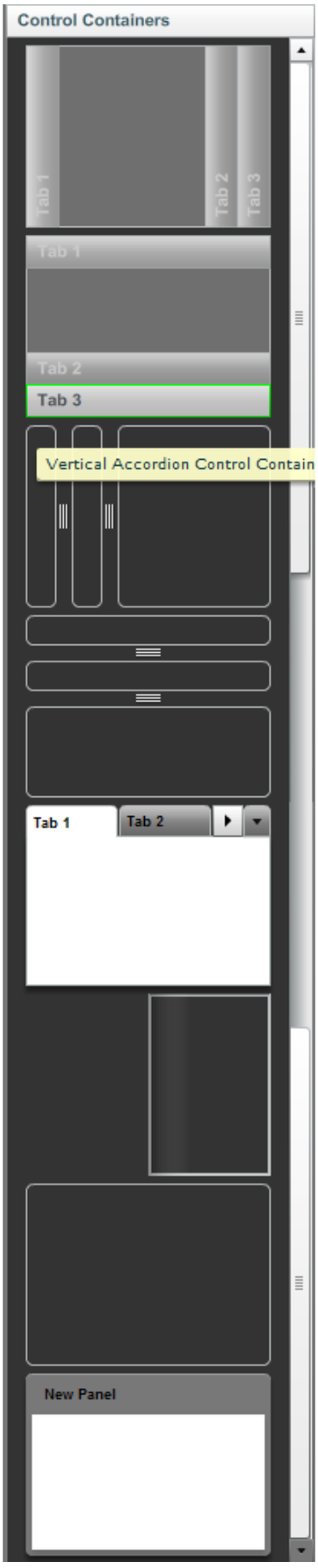
Todos estos controles comparten las propiedades, métodos y eventos de la interfaz IxpertControl, los marcados con [*] implementan además la interfaz IxperValueControl y los marcados con [+] la interfaz IxpertDataControl

Para ver el detalle de propiedades, métodos y eventos de cada uno de estos controles vea la sección Anexo Clases de Controles

Con los controles compuestos puede darle un aspecto mucho más impactante al frontend de su aplicación. Este set de componentes le abre puertas a repensar sus aplicaciones en nuevas dimensiones, desde el manejo de contenidos CMS, integración con servicios de información geográfica, presentación de información grafica y otras facilidades impactantes.

Por otro lado, como composición de otros controles, encapsulan una funcionalidad específica ahorrándole asi el esfuerzo de tener que programarla desde cero combinando los demás controles y programando toda la lógica de eventos particular.

Control Containers



Dentro de los contenedores (en el grupo correspondiente de la toolbox) puede encontrar los siguientes controles:

- Acordion Horizontal (xpertAccorionH) [+]
- Acordion Vertical (xpertAccordionV) [+]
- Divisor Horizontal (xpertDividerH) [+]
- Divisor Vertical (xpertDividerV) [+]
- Solapas (xpertTabs) [+]
- Libro (xpertBook) [+]
- Caja (xpertBox)
- Ventana flotante (xpertPanel)

Todos estos controles implementan las siguientes interfaces:

- IxpertControl
- IxpertContainer

Los marcados con [+] implementan además la siguientes interfaces:

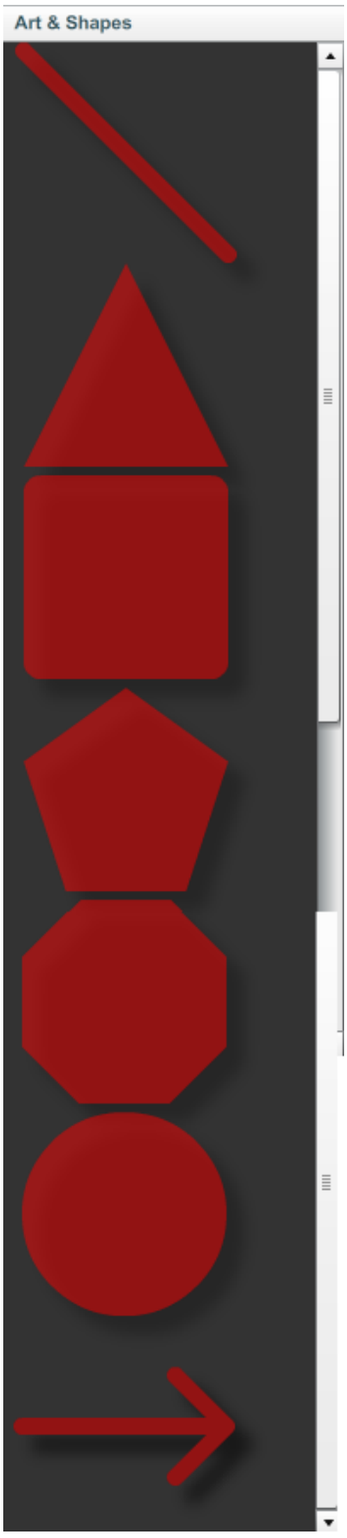
- IxpertMultiContainer
- IxpertDataControl

Para ver el detalle de propiedades, métodos y eventos de cada uno de estos controles vea la sección Anexo Clases de Controles

Con los contenedores Uds. puede organizar la navegación entre los datos así como también modularizar su aplicación dividiéndola en secciones cargables dinámicamente.

La estructura de manejo de datos de los multi contenedores es tal que habilitan asignar a cada contenedor una referencia a otro formulario, la cual será evaluada y resuelta en tiempo de ejecución, de esta manera pude dividir el proyecto de diseñar la interfaz de su aplicación en unidades mas pequeñas, rápidas de desarrollar y de mantener por separado, permitiendo adicionalmente el re-uso de partes en otros formularios. De esta forma pude distribuir mejor el trabajo entre distintos diseñadores y programadores e integrarlo de forma fácil y rápida, el modo preview del designer le permite ver cómo va a lucir y operar su aplicación en tiempo de ejecución y la performance de carga y presentación de las secciones en tiempo de ejecución es increíble.

Art &Shapes



Dentro de los controles de dibujado (en el grupo correspondiente de la toolbox) puede encontrar los siguientes controles:

- Líneas (xpertLine)
- Triángulos (xpertTriangle)
- Cuadrados (xpertSquare)
- Pentágonos (xpertPentagon)
- Hexágonos (xpertHexagon)
- Círculos (xpertCircle)
- Flechas (xpertArrow)

Todos estos controles comparten las propiedades, métodos y eventos de las interfaces IxpertControl y IxpertContainer por lo que pueden operar como contenedores de otros controles demás de ser incluidos dentro de la pantalla y otros contenedores (como ser ellos mismos).

Para ver el detalle de propiedades, métodos y eventos de cada uno de estos controles vea la sección Anexo Clases de Controles

Con los controles de dibujado podrá adornar su interface para hacer más fácil y agradable de entender la información presentada. Haciendo uso de la propiedad “rotation” podrá ubicar el dibujo en la posición que les resulte más útil.

Deje fluir su beta artista (o la de sus diseñadores) enriquezca el frontend de sus aplicaciones con vistosos cuadros, esquemas i diagramas utilizando y combinando este set de controles de dibujado.

Combinando estos controles con los controles de datos de los otros set lograra resultados impactantes y sin sacrificar performance de la aplicación o aumentar los tiempos de desarrollo de la GUI de sus aplicaciones.

La Ventana de Propiedades

La ventana de propiedades le permite ver y modificar las propiedades básicas de los controles, para esto, basta con seleccionar un control de su formulario (aparecerán unos grab handlers) y en la designer window seleccionar la solapa de propiedades "Properties". Al hacer esto, aparecerán las propiedades de dicho control, cada control tiene sus propiedades especificas, puede ver el detalle de cada uno en el Anexo Clases de Controles, pero a sus ves, todos los controles implementan diversas interfaces que se caracterizan por tener ciertas propiedades comunes, puede ver esto en la sección Jerarquía de Interfaces.

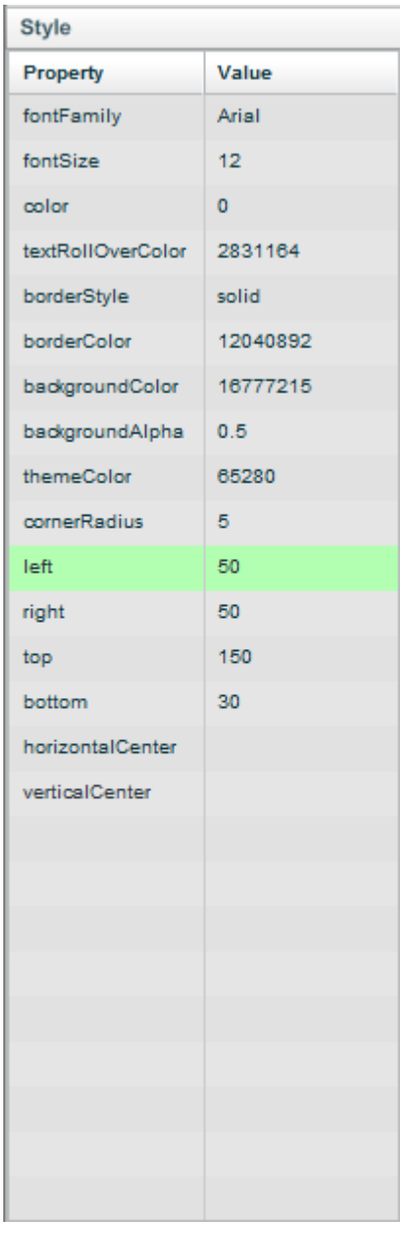
Si ha seleccionado más de un control (presionando la tecla CTL mientras presiona el botón derecho del mouse sobre un control), al cambiar la propiedad en esta ventana estará modificándola a todos los controles seleccionados (lo que simplifica ciertas tareas repetitivas).

Entre las propiedades más comunes que puede administrar aquí, se encuentra las de la interfaz IxpertControl, que entre otras incluyen:

- id: Identificador del control
- x: posición del control en el eje x (en pixels)
- y: posición del control en el eje y (en pixels)
- enabled: Indicador de habilitado para interacción
- visible: Indicador de visibilidad del control
- styleClass: clase de la hoja de estilos dinámica

Además de poder manejar las propiedades en tiempo de diseño desde esta ventana de propiedades, las mismas pueden ser modificadas en runtime mediante el uso de scripts para eventos (vea La Ventana de Eventos).

La Ventana de Propiedades de Estilo



La ventana de propiedades de estilo le permite ver y modificar las propiedades de estilo de los controles, para esto, basta con seleccionar un control de su formulario (aparecerán unos grab handlers) y en la designer window seleccionar la solapa de propiedades de estilo "Style". Al hacer esto, aparecerán las propiedades de estilo de dicho control, cada control tiene sus propiedades de estilo especificas, puede ver el detalle de cada uno en el Anexo Clases de Controles, .

Si ha seleccionado más de un control (presionando la tecla CTL mientras presiona el botón derecho del mouse sobre un control), al cambiar la propiedad en esta ventana estará modificándola a todos los controles seleccionados (lo que simplifica ciertas tareas repetitivas).

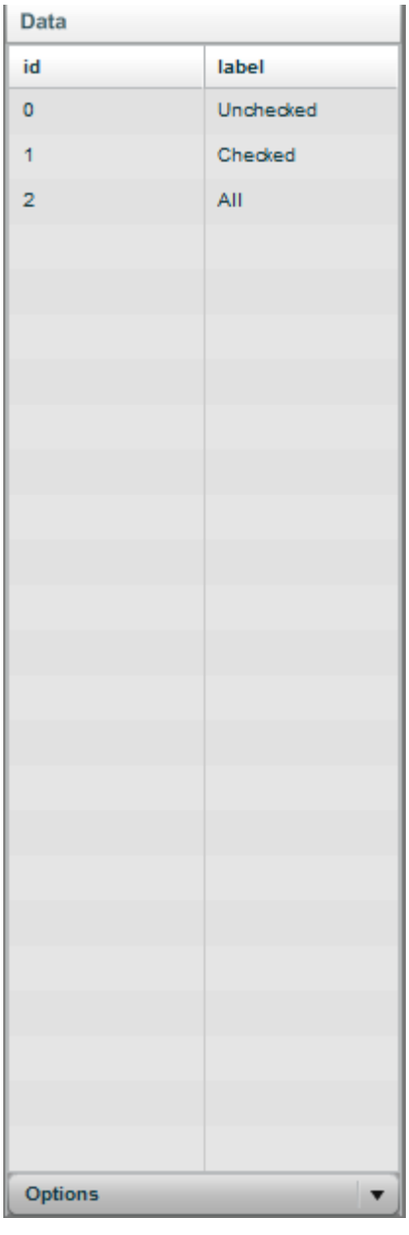
Entre las propiedades de estilo más comunes que puede administrar aquí, se encuentra las siguientes:

- fontFamily: tipo de letra
- fontSize: tamaño de letra
- color: color de letra
- backgroundColor: color de fondo
- backgroundAlpha: % de transparencia del fondo
- borderStyle: tipo de borde

Además de poder manejar las propiedades de estilo en tiempo de diseño desde esta ventana de propiedades, las mismas pueden ser modificadas en runtime mediante el uso de scripts para eventos (vea La Ventana de Eventos).

NOTA: si utiliza la propiedad **styleClass** de los controles para signar un estilo dinámico (administrado con el Editor de Hojas de Estilo) entonces no aplican estas propiedades de estilo y el estilo de los controles se determinara en tiempo de ejecución según el tema elegido por el usuario final.

La Ventana de Datos



Los controles que implementan la interfaz IxpertDataControl soportan configuración de datos internos al control (combos, grillas, listas, arboles, son un ejemplo de ellos).

Esos datos, pueden ser generados en tiempo de diseño y modificados en la ventana de datos, o en tiempo de ejecución desde scripts de eventos o por binding de datos a web services (resueltos en el backend, capa de reglas de negocios y acceso a datos) o una mezcla de ambos.

Este tipos de controles, pueden tener uno o más conjuntos de datos (Datasets) asociados (por ejemplo las grillas tienen un conjunto de datos para describir las columnas y otro para describir los datos de las filas). Los conjuntos de datos se muestran en el combo o lista desplegables que aparece debajo de la tabla de datos, seleccionado los distintos conjuntos de datos se obtiene acceso a cada uno de ellos para modificarlos.

En la grilla de datos puede modificar cualquier dato presentado alterando de esta forma el contenido del control seleccionado.

Para generar estos datos en tiempo de diseño se suelen usar las opciones del menú contextual del control, por ejemplo la opción de menú “Add Option” en combos y listas.

Para ver como generar y administrar los datos en tiempo de ejecución vea la sección Ventana de Eventos

Nota: Los multi contenedores también tienen soporte de datos, allí los datos determinan la cantidad de contenedores del multi contenedor.

La Ventana de Eventos

Los eventos le permiten asociar una secuencia de comandos o acciones a realizar como respuesta a cierto hecho ocurrido en la interfaz del usuario, como puede ser, que el usuario presione el botón del mouse sobre cierto control (evento “click”).

Para editar los eventos de un control debe abrir el outliner (en la opción de menú **View / Outliner** de la barra de menú del designer, *ver Fig. 1*), luego seleccionar el control (ya sea haciendo click sobre el propio control o sobre el nodo correspondiente al control en el tree de controles del outline, *ver Fig. 2*), luego seleccionar el evento en la lista de eventos del control que aparecen arriba a la

derecha del outliner, escribir o hacer los cambios deseados al script y por ultimo presionar el botón de salvar cambios para registrar en el evento el nuevo script (*ver Fig. 3*).

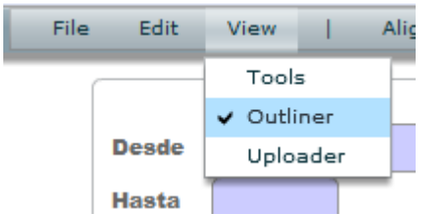


Fig. 1 – Menu **View / Outliner** del Designer

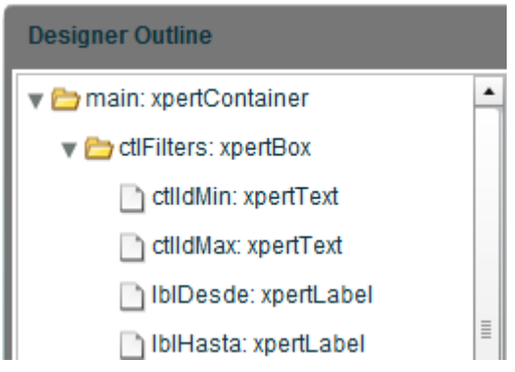


Fig 2. – Outline de controles del Designer

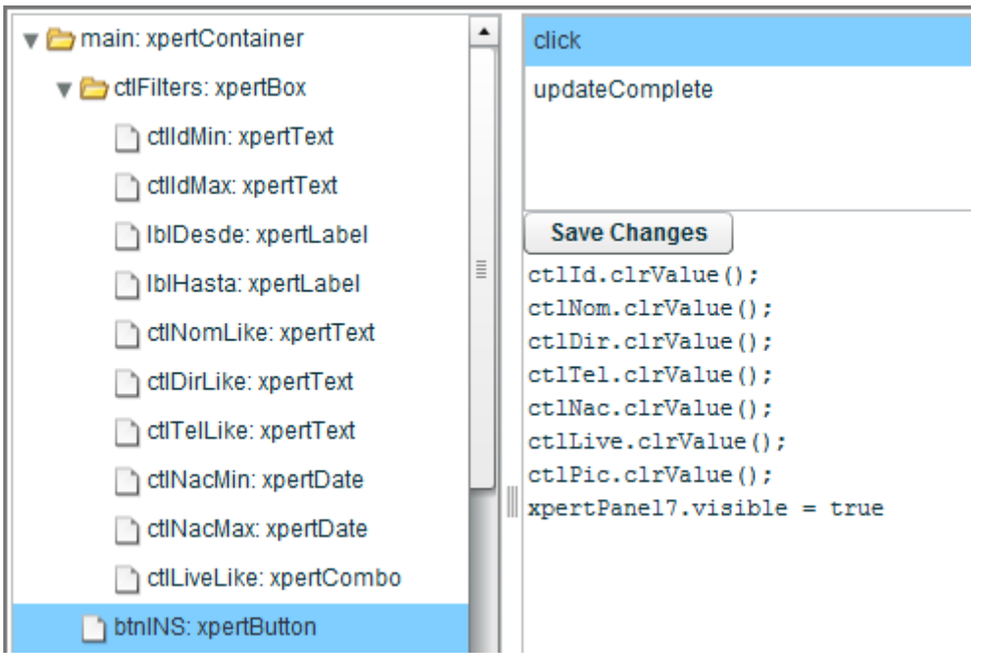


Fig 3. – Eventos de un determinado control en el Outliner.

La sintaxis que se utiliza para escribir los eventos es básicamente la de JavaScript o ActionScript (ECMA 262 en ambos casos), dispone de los siguientes elementos (para ver un detalle completo vea el Anexo Sintaxis, debe tener especial cuidado porque la sintaxis es **case sensitive**):

```
if (<Expresión>) {
        <Bloque de Sentencias>
} else if (<Expresión>) {
        <Bloque de Sentencias>
} else {
        <Bloque de Sentencias>
}
```

```
while (<Expresión>) {
        <Bloque de Sentencias>
}
```

```
for (<Sentencia>; <Expresión>; <Sentencia>) {
        <Bloque de Sentencias>
}
```

```
for (<Variable> in <Expresión>) {
        <Bloque de Sentencias> // Típico acceso: <Expresión>[<Variable>]
}
```

```
for each (<Variable> in <Expresión>) {
        <Bloque de Sentencias> // Típico acceso: <Variable>.<Método o Propiedad>
}
```

El objeto CONTROLLER y otros parámetros en Eventos

Dentro del código ECMA-262 que Ud. Puede asociar a los eventos, dispone de ciertos objetos predefinidos, recibidos como parámetro, para facilitar su codificación, ellos son:

- control
- event
- item
- controller

El objeto "**control**" representa el control en el cual se disparo el evento, Ud. Puede consultar o asignar cualquier propiedad o invocar cualquier método del control mediante la construcción:

control.<propiedad> o control.<método>(<lista de parámetros>)

Adicionalmente, puede referenciar a cualquier control ubicado en la pantalla directamente por su nombre (asignado en la propiedad "id" de la ventana de propiedades), de esta forma, también puede utilizar esta construcciones equivalente a las previas pero para un control específico:

<id control>.<propiedad> o <id control>.<método>(<lista de parámetros>)

El objeto "**event**" tiene diferentes propiedades acerca del evento que ha ocurrido. Según el tipo de evento, puede contener coordenadas de un click, o la tecla presionada u otros datos informativos del evento.

El objeto "**ítem**", es de utilidad cuando el evento ocurre en alguno de los controles que implementan la interfaz IxpertDataControl, en dicho caso (estos controles presentan visualmente una colección de ítems basados en datos), este parámetro contiene el ítem (Ej.: nodo de un árbol o una lista, fila de un grilla o subformulario de un Tile) sobre el cual a ocurrido el evento.

Por último, dispone del objeto "**controller**" el cual le permite realizar operaciones globales sobre los controles así como todas las operaciones de comunicación con el backend, como ser, enviar y recibir datos y/o formularios.

El objeto controller dispone de las siguientes propiedades:

menuBar	designerMenuBar	Menú del designer, null en runtime
toolBox	designerToolBar	Barra de herramientas del designer, null en runtime
output	designerOutput	Ventana de salida XML del designer, null en runtime
uploader	xpertUploader	Ventana de upload de archivos
outliner	designerOutline	Ventana de edición de eventos del designer, null en runtime
main	xpertContainer	Control contenedor principal
fileName	String	Nombre de archivo del formulario cargado en el objeto main
clipboard	DisplayObject	Objeto temporal para operaciones de cut&paste del designer
backup	XML	Respaldo de metadatos para recuperación luego de un preview
data	Object	Espacio auxiliar de uso genérico para almacenar información global
selectedControls	Array	Lista de controles seleccionados en el designer
controlCounter	Number	Cantidad de controles creados
mode	String	Modo (Diseño, Ejecucion, etc.)
version	String	Versión del producto
config	XML	contenido del archivo de configuracion
rootFolder	String	atributo root del loader en el config.xml
assetsFolder	String	atributo root del uploader en el config.xml
loaderup	String	atributo url del uploader en el config.xml
loader	String	atributo url del loader en el config.xml
bridge	String	atributo url del bridge en el config.xml

Y los siguientes métodos:

rootUrl():String	Ruta donde esta corriendo el producto
product():String	Nombre del producto (designer, runner, Styler, etc)
getNextId(t:String):String	Proximo numero de control a crear
DTS(d:Date):String	Convierte fechas a string en formato YYYYMMDD
STD(d:String):Date	Convierte strings en formato YYYYMMDD a fecha
refreshFormat(o:IxpertControl):void	Aplica el estilo dinámico seleccionado al control indicado
loadDefinition(f:String):void	Carga en el objeto main un formulario de la ruta f
saveDEfinition(f:String):void	Salva el contenido del objeto main en la ruta f

selectFile(f:String,t:String,Fn:Function):void Abre el dialogo para seleccionar archivos
findChild(o:IxpertControl,c:String):IxpertControl Busca un control por su nombre dentro de un contenedor
getObjects(o:IxpertControl,x:Object,p:*='):void Devuelve un objeto con los valores de IxpertValueControls contenidos en el contenedor o indexado por la propiedad bindName de dichos controles
gxEvent(e:String,d:Boolean=false):void Dispara un evento Genexus v8/v9 cuando el formulario esta embebido en un webpanel Genexus
gxData(v:String,p:String,s:String,d:Boolean=false):void Asigna un valor a un control Genexus v8/v9 cuando el formulario esta embebido en un webpanel Genexus
jsGet(C:String,P:Strinf):String devuelve el valor de la propiedad "p" del control "c" del HTML container
jsSet(C:String, P:String,V:String):void modifica el valor de la propiedad "p" del control "c" del HTML container
jScript(S:String,W:String='_self',d:Boolean=false):void ejecuta un script JavaScript en el HTML container
notify(e:Event=null):void Notifica la ocurrencia de un evento al HTML container
notifyEvent(c:String, e:String, i:Object):void Idem
listener(s:String, p:Object=null):Function Devuelve una function en base al codigo action script
success(s:String):Function DEvuelve una function valida para los onSucces de los metodos call, getData, getForm, getXML
fault(s:String):Function DEvuelve una function valida para los onError de los metodos call, getData, getForm, getXML
encrypt(m:String, k:String):String Encripta el mensaje "m" con la clave "k"
decrypt(m:String, k:String):String Desncripta el mensaje "m" con la clave "k"
executeScript(c:IxpertControl,s:String,e:Event=null,p:Object=null): void ejecuta el script "s" en el ambito del control "c"
exeScript(e:Event=null):void Ejecuta el script asociado al evento "e" en el target del evento
execute(ctl:*, hdl:String, act:String, par:*, trg:*, data:*):void
getControl(c:*):IxpertControl Busca el control por su nombre
getCookie(s:String):String Devuelve el valor de la cookie "s"
setCookie(s:String, v:String):String Asigna el valor "v" a la cookie "s"
getGlobal(s:String):String Devuelve el valor de la variable global "s"
setGlobal(s:String, v:String):String Asigna el valor "v" a la variable global "s"
getParameter(s:String):String Devuelve el valor del parametro "s" pasado por GET o FlashVars
getName(o:*):String Devuelve el nombre del control "o"
getLink(o:*):String Busca la propiedad "url" dentro del objeto o XML
getBinds(c:IxpertControl=null):void Construye un objeto con todos los binds del contenedor "o"
call(h:String, p:*=null,S:*=null,E:*=null):void Invoca al procedimiento remoto "h" psandole los parametros "p"
getData(h:String,t:*,p:*='',S:*=null,E:*=null):void Idem, pero los datos retornados los usa para cargarlos en el conrol "t"
getForm(h:String,t:*,p:*='',m:String='R',I:String='', S:*=null,E:*=null):void Idem, pero para cargar un formulario en el contenedor "t"
getXML(H:String,M:XML,E:Function,S:Function):void Hace una llamada remota asando un mensaje XML, fija las retinas E y S para el caso que falle el RPC o tenga exito
animate(control:IxpertControl, property:String, targetValue:*, time:Number):void realize animaciones sobre propiedades de l control

Detallamos a continuación los métodos más comúnmente usados en sus eventos:

```
controller.call(<handler>, [, <parameters> [,<onSucces> [, <onError>]]]);
controller.getData(<handler>, <target> [, <parameters> [,<onSucces> [, <onError>]]]);
```

```
Ej: controller.getData('ppeoplelst',migrilla, {desde: 0, hasta: 99, pageSize: ctlPageSize})
```

donde:

1. <handler> es un String que contiene el nombre del procedimiento Genexus a invocar
2. <target> puede ser o bien un string con el nombre del control (por ej: 'myGrid') donde se va a cargar el artefacto o directamente el control, por ej: myGrid. <target> debe ser algún control de los que implementan la interfaz IxpertDataControl (que implementan el método loadData, combos, grillas, tres, etc.) o puede ser null en el caso de planear procesar los resultados por código en lugar de enviarlos directo a que los muestre un control.
3. <parameters> puede ser o bien un string con la lista de parámetros separados entre "," y opcionalmente separando el nombre del parámetro del nombre del control con ":" o un objeto cuyas propiedades son el nombre de parámetro y el valor para el parámetro:

'<ctlID>,…'
EJ: 'control1,control2,control3' (valido si los parámetros del proc Gx llevan el mismo nombre que el ID de los controles que se desean enviar como valor)
'<parID>:<ctlID>,…'
EJ: 'p1:control1,p2:control2,p3:control3' (aquí se indica para cada nombre de parámetro el nombre del control de donde saldrá el valor a enviar para el mismo)
{<parID> : <parValue>,…}
Ej: {p1: 100, p2: 'SP', p3: controller.getControl('usr'), p4: controller.getCookie('userpref1') }

Los <parID> son nombres arbitrarios, pero son el punto de coordinación entre la GUI y el proc remoto (por ej: GX), dicho de otra forma, los nombres con los que el proc busque los parámetros en el inSDT, deben ser los nombres que se usen en la lista <parametrs> para enviar los valores al proc.

En las opciones donde la lista de parámetros en usada como un string de nombres separados por coma 'control1,control2,control3', se esta asumiendo lo siguiente:

- La lista está determinando los nombres de los controles de la GUI que contienen los valores que se desean enviar como parámetro al proc GX
- Los nombres de cada uno de esos controles coinciden con el nombre de parámetro con que el proc GX consulta por cada parámetro (en el proc GX se buscan los parámetros control1, control2, control3)
- Todos los controles usados como parámetro son de algún tipo de control que implemente la interfaz IxpertValueControl (tienen los métodos clrValue, getValue y setValue para obtener el valor del control)

En el caso de que la lista de parámetros sea un string de la forma 'p1:control1,p2:control2,p3:control3', valen las mismas consideraciones que antes, pero se establece un mapeo de controles a nombres de parámetro, por tanto:

- Los controles control1, control2, control3 son los que contienen los valores a enviar al proc GX, pero los nombres de parámetro con que se deben obtener los valores en el proc GX son p1, p2, p3

En el caso de usar el esquema de envío de parámetros usando un objeto de la forma {p1: 100, p2: 'SP', p3: controller.getControl('usr'), p4: controller.getCookie('userpref1') } vale lo siguiente:

- Los nombres de las propiedades del objeto serán usados como los nombres de los parámetros con que se buscan los valores en el proc GX, p1, p2, p3, p4
- Los valores a enviar como parámetros puede ser cualquier expresión, por ejemplo un string literal, una constante, un control IxpertValueControl (en cuyo caso lo que se envía es el resultado del método getValue) o la cualquier propiedad de un control referido como control.propiedad, o inclusive expresiones de cálculo sobre combinaciones de las anteriores.

Los parámetros opcionales *onSucces* y *onError* son strings que te permiten dar scripts alternativos a ejecutar en caso de que la llamada RPC sea exitosa (para hacer un tratamiento distinto al por defecto que consiste en cargar los datos en un control IxpertDataControl) o que sea fallida (para dar un tratamiento distinto al por defecto para el caso de fallar la llamada RPC, que consiste en dar un mensaje de alerta con la descripción del error)

el objeto controller te ofrece además, un método para generar/obtener pantallas (formularios) generados por código (como alternativa a las pantallas armadas con el designer y salvadas en el repositorio) , la sintaxis del método getForm es la siguiente:

```
controller.getForm(<handler>, <target> [, <parameters> [, <mode> [, <artifact> [, <onSucces> [, <onError>]]]]]);
```

donde:

1. <handler> es un String que contiene el nombre del procedimiento Genexus a invocar salvo que <mode> valga “R” en cuyo caso contiene la URL del artefacto a cargar desde el repositorio
2. <target> puede ser o bien un string con el nombre del control (por ej: ‘main’) donde se va a cargar el artefacto o directamente el control, por ej: controller.getControl(‘main’)
3. <parameters> puede ser o bien un string con la lista de parámetros separados entre “,” y opcionalmente separando el nombre del parámetro del nombre del control con “:” o un objeto cuyas propiedades son el nombre de parámetro y el valor para el parámetro:
   ‘<ctlID>,…’

   EJ: ‘control1,control2,control3’ (valido si los parámetros del proc Gx llevan el mismo nombre que el ID de los controles que se desean enviar como valor)
   ‘<parID>:<ctlID>,…’
   EJ: ‘p1:control1,p2:control2,p3:control3’ (aquí se indica para cada nombre de parámetro el nombre del control de donde saldrá el valor a enviar para el mismo)
   {<parID> : <parValue>,…}
   Ej: {p1: 100, p2: ‘SP’, p3: controller.getControl(‘usr’), p4: controller.getCookie(‘userpref1’) }

4. <mode> es un carácter que puede tomar alguno de los siguientes valores (el valor por defecto de este parámetro es “R”):
   - “R”: significa que el formulario a mostrar dentro del control <target> es el resultado de cargar el artefacto ubicado en la ruta recibida en el parámetro <hanlder>, ejemplos de uso de este modo pueden ser los siguientes (basta con indicar solo los 2 primeros parámetros ya que los restantes son opcionales y no son necesarios en este modo):

     ```
     controller.getForm(‘artifacts/site/Productos/productos.xml’, main);
     o
     controller.getForm(‘artifacts/site/Productos/productos.xml’, ’main’);
     o
     var trg = controller.getControl(’main’); controller.getForm(‘artifacts/site/Productos/productos.xml’, trg);
     ```

   - “G”: significa que el formulario a mostrar dentro del control <target> es 100% generado por el proc Gx <hanlder> (para esto, el proc Genexus debe recibir un SDT de tipo xpertParms y retornar un SDT de tipo xpertControls en lugar del típico xpertResult), en este caso tiene sentido el uso de los 3 primeros parámetros:

     ```
     controller.getForm(‘pgenfrm’, ‘main’, ‘ctl1,ctl2’); //suponiendo que los ID de los controles coincidan con los nombres de parámetros esperados en el proc GX
     o
     var trg = controller.getControl(’main’); controller.getForm(‘pgenfrm’, trg, ‘p1:ctl1,p2:ctl2’); //suponiendo un mapeo de ID de control a nombre de parámetro
     o
     var par = { p1: controller.getControl(‘ctl1’), p2: controller.getControl(‘ctl2’)}; controller.getForm(‘pgenfrm’, ‘main’, par);
     o
     var ctl1 = controller.getControl(‘ctl1’); var ctl2 = controller.getControl(‘ctl2’); var par = { p1: ctl1, p2: ctl2 }; controller.getForm(‘pgenfrm’, ‘main’, par);
     ```

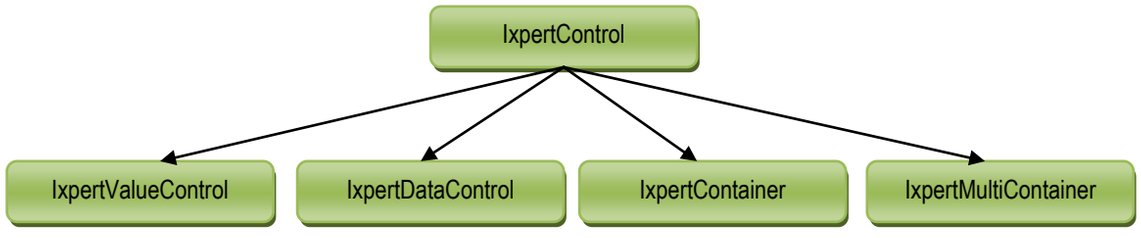
o
controller.getForm('pgenfrm', 'xpertDivider7.TAB2', {p1: 10, p2: 'SP'}); //usando constantes como parámetros

- "T": significa que el formulario a mostrar dentro del control <target> es una transformación de un formulario enviado como parámetro (útil por ej. para traducciones). Se utilizan normalmente los primeros 3 parametros. En este caso, el tanto el inSDT, como el outSDT del proc Gx deben ser de tipo xpertControls.
- "D": significa que el artefacto indicado por la ruta <artifac> solo será cargado en el control <target> si luego de ejecutar el proc Gx <handler> este retorna STATUS="OK", si en cambio el proc Gx devuelve STATUS="ERROR" se despliega el mensaje MSG y no se carga el formulario. Basicamente es una carga condicional determinada por la propiedad STATUS del SDT de salida del proc gx <hanlder>

5. <artifact> es un parámetro que solo es usado cuando <mode> = "D", y le indica la URL al artefacto a cargar en <target> si la ejecución del proc Gx <handler> devuelve STATUS="OK"

Jerarquía de Interfaces

En el código de los eventos pude manipular directamente los distintos controles dispuestos en la pantalla (estén o no visibles) mediante su identificación (propiedad "id" del control), pudiendo utilizar cualquier propiedad o método del control (*ver Fig. 3*). Cada clase de controles dispone de distintas propiedades y métodos (vea un detalle completo en el anexo Clases de Controles). Para simplificar el manejo y compresión de los mismos, toda clase implementa varias interfaces que describen las propiedades y métodos comunes. El siguiente diagrama muestra la jerarquía de herencia entre dichas interfaces:



IxpertControl

Caracteriza mínimamente a la totalidad de controles ofrecidos por xperTools, incluye las siguientes propiedades, estilos y métodos:

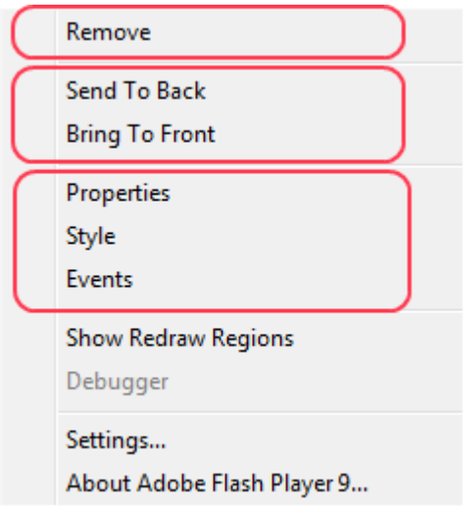
Propiedad	Tipo	Restricción	Descripción
id	String		identificador del control
bindName	String		Nombre de parámetro RPC para binding
className	String	Readonly	nombre de la clase del control
x	Number		posición x en pixels
y	Number		posición y en pixels
width	Number		ancho en pixels del control
height	Number		altura en pixels del control
tabIndex	Number		Orden de tabulación de controles
rotation	Number	0 a 360	ángulo de rotación sobre el plano x, y
visible	Boolean		visible
enabled	Boolean		habilitado
toolTip	String		descripción emergente
parent	IxpertContainer		contenedor
contextMenu	ContextMenu		Menú contextual
propes	Array		Lista de nombres de propiedad aceptadas
styles	Array		Lista de propiedades de estilo aceptadas
events	Object		Mapeo de nombres de evento a Scripts
styleClass	String		Nombre de clase de estilo del Styler

Estilo	Tipo	Restricción	Descripción
backgroundAlpha	Number	0..1	Transparencia del fondo
backgroundColor	Number	0x000000..0xFFFFFF	Color del fondo
borderStyle	String	none, solid, dashed	Tipo de borde
borderThickness	Number		Grosor del borde
borderColor	Number	0x000000..0xFFFFFF	Color del borde
fontFamily	String		Tipo de letra
fontSize	Number		Tamaño de letra
fontWight	String	bold	Negritas
fontStyle	String	Italic	Itálicas
fontDecoration	String	underline	Subrayado
color	Number	0x000000..0xFFFFFF	Colo de letra
disabledColor	Number	0x000000..0xFFFFFF	Color de letra deshabilitado
rollOverColor	Number	0x000000..0xFFFFFF	Color de letra mouse sobre
themeColor	Number	0x000000..0xFFFFFF	Color de tema
textAlign	String	left, center, right	Alineación del texto
cornerRadius	Number		Radio de esquinas (redondeado)

Método	Retorno	Parámetros	Descripción
toXML	XML		metadata del control
fromXML	void	metadata:XML	Crea desde metadata
toString	String		
getStyle	*	propiedad:String	Devuelve estilo
setStyle	void	propiedad:String,valor:*	Asigna estilo
clone	IxpertControl		Clona el control
remove	void		Elimina el control
sendToBack	void		Envía el conrol al fondo (z-order)
bringToFront	Void		Trae el control al frente (z-order)

Evento	Descripción
creationComplete	Se dispara cuando el control se termina de construir, ideal para llamadas al método loadFrom para cargar datos en controles IxpertDataControl
updateComplete	Se dispara cuando un control es redibujado por algún motivo, ideal para resizing
mouseOver	Se dispara cuando el cursor entra al espacio ocupado por el control
mouseOut	Se dispara cuando el cursor sale del espacio ocupado por el control

Todos los controles que implementan la interfaz IxpertControl, tienen al menos las siguientes opciones de menú contextual:



Remove: permite eliminar el control

Send To Back: envía el control hacia atrás
Bring To Front: trae el control hacia adelante

Properties: abre la ventana de propiedades
Style: abre la ventana de propiedades de estilo
Events: abre la ventana de eventos

IxpertValueControl

Caracteriza a los controles que manejan un valor asignable (Checkbox, Imagen, Cuadro de Texto, etc.). Extiende IxpertControl con las siguientes propiedades y métodos:

Propiedad	Tipo	Restricción	Descripción
bindName	String		Nombre de parámetro RPC para binding

La propiedad "bindName" permite simplificar el envio de parámetros en las llamadas a controller.call, controller.getData, controller.getForm asi como los métodos load de los IxpertContainer y loadFrom de los IxpertDataControl, a modo de ejemplo:

Puede escribir:

```
miGrilla.loadFrom('miProcRemoto');
o
miGrilla.loadFrom('miProcRemoto', controller.getBinds(contenedorDeParametros));
```

en lugar de esto:

```
var parm = {    p1: ctl1.getValue(),
                p2: ctl2.getValue(),
                …
                pn: ctln.getValue() }
miGrilla.loadFrom('miProcRemoto', p);
```

Asignandole el nombre de parámetro a la propiedad bindName de los controles que desea enviar al procedimiento remoto.

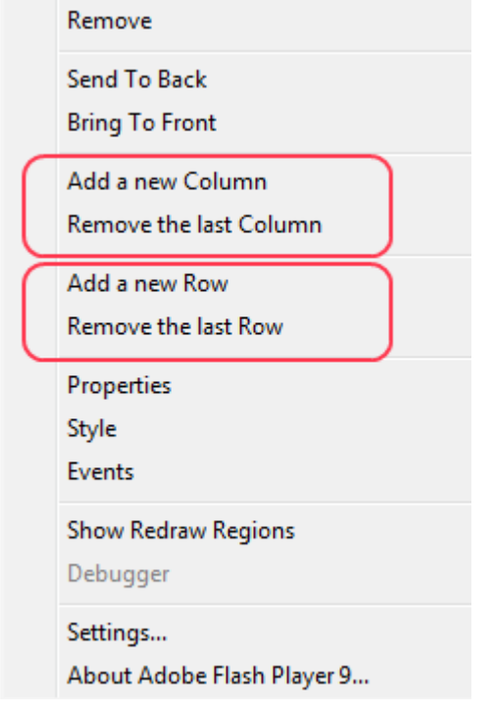
Método	Retorno	Parámetros	Descripción
clrValue	void		resetea el valor del control
getValue	*		devuelve el valor del control
setValue	void	valor:*	asigna el "valor" al control

IxpertDataControl

Caracteriza a los controles que manejan un contenido que puede ser cargado desde datos externos (List, Tree, etc.). Extiende IxpertControl con los siguientes métodos:

Método	Retorno	Parámetros	Descripción
checkForData	void		Dispara el evento datos
loadData	*	Data:XML	Carga el XML dentro del control
loadFrom	void	proc:String, par:Object	Carga el control con los datos devueltos por el "proc" remoto

Los controles que implementan la interfaz IxpertDataControl agregan a su menú contextual las opciones para agregar y quitar datos, un par (Add/Remove) para cada DataSet manejado. Por ejemplo, las grillas tienen un DataSet para las columnas y otro para las filas:



Add/Remove dataset de columnas

Add/Remove dataset de filas

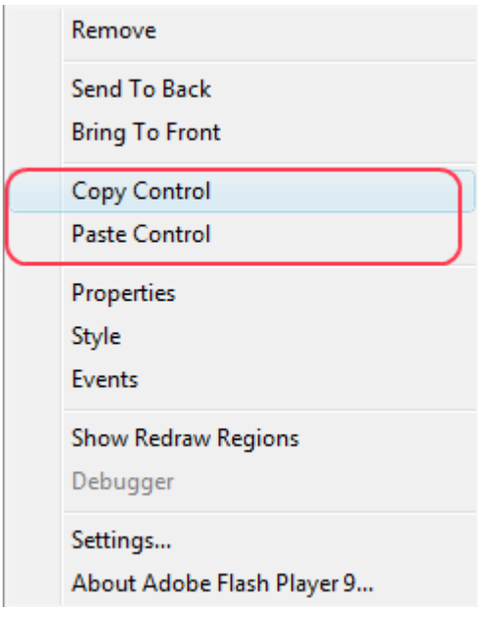
IxpertContainer

Caracteriza a los controles que pueden contener controles dentro (Container, Panel, Box, etc.). Extiende IxpertControl con las siguientes propiedades y métodos:

Propiedad	Tipo	Restricción	Descripción
numChildren	Number		Cantidad de controles contenidos

Método	Retorno	Parámetros	Descripción
clear	void		Dispara el evento datos
addChild	IxpertControl	child:IxpertControl	Agrega el control al contenedor
getChildAt	IxpertControl	Position:Number	Devuelve el control contenido
removeChildAt	IxpertControl	Position:Number	Elimina el control contenido
clrChildrenValues	void		Vacia los valores de los controles
getChildrenValues	Object		Devuelve un objeto con los valores
setChildrenValue	Void	Values:Object	Asigna los valores a los controles
load	void	url:String	Carga artefacto en contenedor
save	void	uri:String	Salva el artefacto en la dirección uri

Los controles que implementan la interfaz IxpertContainer, agregan a su menú contextual opciones para hacer cut &paste de controles, habilitando mediante ellas la posibilidad de transferir un control de un contenedor a otro.



Copy: copia el control seleccionado al clipboard
Paste: Crea un control desde el clipboard

IxpertMultiContainer

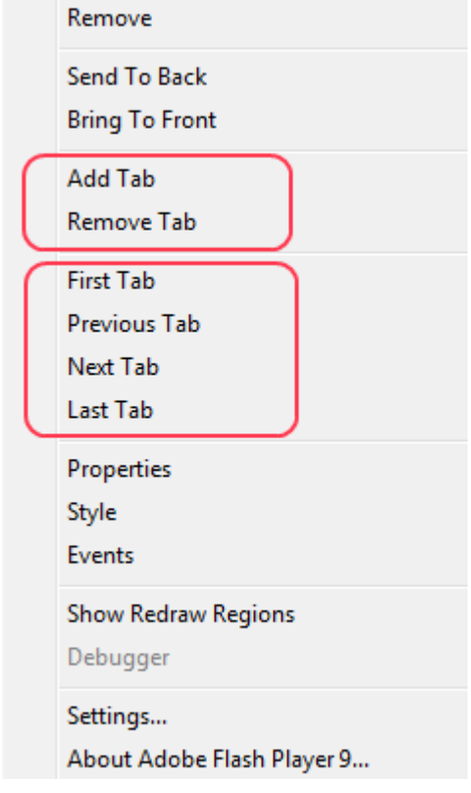
Caracteriza a los controles que permiten navegar entre varios contenedores de controles (Accordion, TabFolder, Divider, Book, etc). Su función es permitir identificar si un determinado controls es en particular un multi contenedor o un contenedor, por lo demás describen las mismas propiedades y métodos que los contenedores.

Propiedad	Tipo	Restricción	Descripción
numChildren	Number		Cantidad de controles contenidos

Método	Retorno	Parámetros	Descripción
clear	void		Dispara el evento datos
addChild	IxpertControl	child:IxpertControl	Agrega el control al contenedor
getChildAt	IxpertControl	Position:Number	Devuelve el control contenido
removeChildAt	IxpertControl	Position:Number	Elimina el control contenido

Todos los hijos directos de un multi contenedor son de tipo **xpertBox**, dentro de cada xpertBox puede colocarse cualquier tipo de control que implemente la interfaz IxpertControl.

Los multicontainers agregan al menú contextual las opciones necesarias para agregar o quitar un sub-contenedor o solapa (xpertBox) asi como opciones para recorrer las distintas solapas



Add: Agrega una solapa (xpertBox) al multicontenedor
Remove: Quita la ultima solapa (xpertBox) agregada

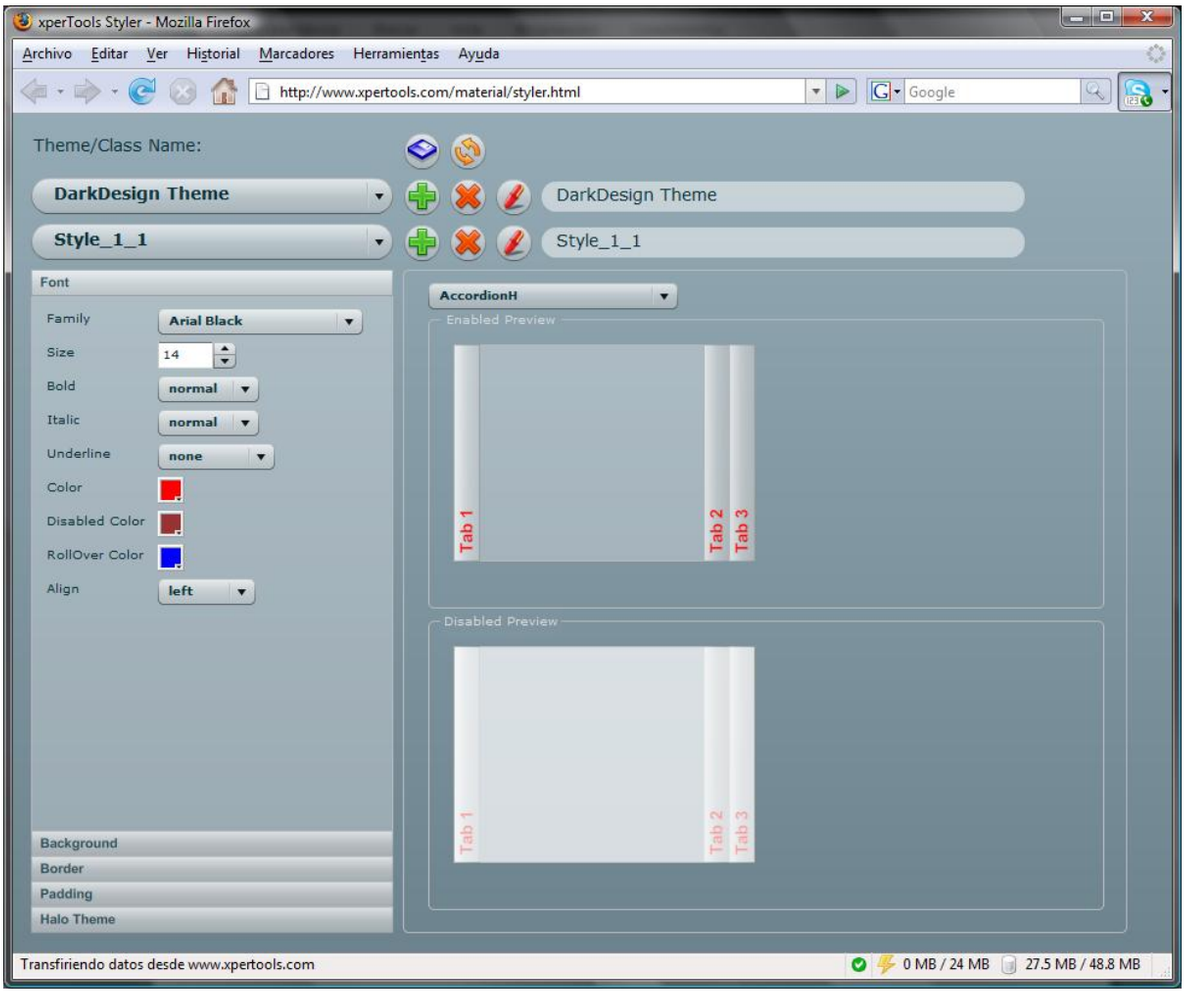
First: se posiciona en la primera solapa
Previous: retrocede una solapa
Next: avanza una solapa
Last: Se mueve a la última solapa

El editor de Hojas de Estilo (Styler)

El editor de hojas de estilo es una herramienta que complementa al designer. Permite editar las hojas de estilo dinámicas usadas por el designer y el runner.

Las hojas de estilo dinámicas, están constituidas por temas (themes) y dentro de cada tema por clases (Style Class). Normalmente, cada clase es redefinida dentro de cada tema.

En tiempo de ejecución, el usuario final de la aplicación puede elegir el tema (alguno de los temas definidos con el Styler) haciendo que las propiedades de estilo de los controles de la GUI se ajusten a las definidas en la hoja de estilo, para esto el desginer y el runner utilizan la propiedad styleClass asociada a cada control. De esta forma, en tiempo de diseño, se le puede asociar a cada control un nombre de clase de estilo que se aplicara posteriormente al ejecutar la aplicación. Esto permite al usuario personalizar el ambiente a su agrado particular.



La herramienta incluye facilidades para seleccionar los temas y dentro de cada tema los estilos, definir, modificar el nombre y eliminar temas y estilos. Modificar las distintas propiedades de estilo (que coinciden con las manejadas para cada control en la ventana de propiedades de estilo del Designer) y ver una presentación preliminar de cada uno de los tipos de control tanto si el mismo está habilitado como deshabilitado.

Anexo Clases de Controles

Solo se indican las propiedades, estilo y métodos especificas de cada control, adicionalmente aplican aquellas inferidas de las interfaces que implementan, ver la sección Jerarquía de Interfaces.

En los casos de controles que implementan la interfaz IxperValueControl se indica que propiedad es manejada por los métodos clrValue, getValue y setValue.

En el caso de le controles que implementan la interfaz IxpertDataControl se indican los juegos de datos administrados.

xpertLabel: IxpertValueControl

El valor de la etiqueta es el texto (text) que muestra.

Propiedades	Estilo	Métodos	Eventos
text			creationComplete updateComplete

xpertLink: IxpertValueControl

El valor del link es la dirección (url) del vínculo al que apunta.

Propiedades	Estilo	Métodos	Eventos
label url			creationComplete updateComplete click

xpertButton: IxpertControl

Propiedades	Estilo	Métodos	Eventos
label icon			creationComplete updateComplete click

xpertCheck: IxpertValueControl

El valor de la casilla de verificación es el estado (marcada/desmarcada) expresado como checkedValue si está marcada y como uncheckedValue en otro caso (selected?checkedValue:uncheckedValue).

Propiedades	Estilo	Métodos	Eventos
label selected checkedValue uncheckedValue			creationComplete updateComplete change click

xpertText: IxpertValueControl

El valor del cuadro de texto es su texto (text).

Propiedades	Estilo	Métodos	Eventos
text			creationComplete
required			updateComplete
requiredMessage			change
validExpresion			enter
invalidMessage			focusOut

xpertNumber: IxpertValueControl

El valor del cuadro numérico es el número (value)

Propiedades	Estilo	Métodos	Eventos
value			creationComplete
minimum			updateComplete
maximum			change
stepSize			

xpertDate: IxpertValueControl

El valor del cuadro de fecha es la fecha seleccionada (text)

Propiedades	Estilo	Métodos	Eventos
text			creationComplete
selectedDate			updateComplete
editable			change
			open
			close

xpertCombo: IxpertValueControl, IxpertDataControl

El valor de la lista de selección es la clave del ítem seleccionado en la lista desplegable

Los datos manejados por la lista desplegable son una lista de OPCIONES, indicando clave y descripción de cada ítem a mostrar en la lista desplegable de opciones.

Propiedades	Estilo	Métodos	Eventos
text		clear()	creationComplete
selectedIndex		addOption(key,label)	updateComplete
ValueField		remOption(index)	change
LabelField			open
items			close

xpertProgress: IxpertControl

Propiedades	Estilo	Métodos	Eventos
mínimum		reset	creationComplete
maximum			updateComplete
			progress
			complete

xpertSliderV: IxpertDataControl

Los datos determinan cada uno de los valores manejados por el múltiple slider.

Propiedades	Estilo	Métodos	Eventos
values		clear()	creationComplete
snapCount		addValue(value)	updateComplete
minimum		remValue(index)	change
maximum		getValue(index)	
snapInterval		setValue(value,index)	
tickInterval			

xpertSliderH: IxpertDataControl

Los datos determinan cada uno de los valores manejados por el múltiple slider.

Propiedades	Estilo	Métodos	Eventos
values		clear()	creationComplete
snapCount		addValue(value)	updateComplete
minimum		remValue(index)	change
maximum		getValue(index)	
snapInterval		setValue(value,index)	
tickInterval			

xpertImage: IxpertValueControl

El valor de la imagen determina la ruta (source) al recurso (BMP, JPG, GIF, SWF) a mostrar.

Propiedades	Estilo	Métodos	Eventos
source			creationComplete
icon			updateComplete
			open
			progress
			complete
			click

xpertClip: IxpertValueControl

El valor de la animación determina la ruta (source) al recurso (SWF) a mostrar

Propiedades	Estilo	Métodos	Eventos
source			creationComplete
icon			updateComplete
			open
			progress
			complete
			click

xpertRadio: IxpertValueControl, IxpertDataControl

El valor del grupo de copiones determina la clave del item seleccionado

Los datos determinan las OPCIONES que integran el grupo

Propiedades	Estilo	Métodos	Eventos
selectedIndex		clear()	creationComplete
ValueField		addOption(key,label,selected)	updateComplete
LabelField		remOption(index)	change
items			click

xpertList: IxpertDataControl

Los datos de la lista identifican los nodos del mismo (clave, descripción y vínculo)

Propiedades	Estilo	Métodos	Eventos
selectedIndex		clear()	creationComplete
ValueField		addItem(key,label,selected,url)	updateComplete
LabelField		remItem(index)	itemClick
editable			itemEditBeginning
allowMultipleSelection			itemEditBegin
dragEnabled			itemEditEnd
dragMoveEnabled			itemFocusIn
dropEnabled			itemFocusOut
items			

xpertTree: IxpertDataControl

Los datos del tree identifican los nodos del mismo (clave, descripción y vínculo)

Propiedades	Estilo	Métodos	Eventos
selectedIndex		clear()	creationComplete
ValueField		addNode(key,label,selected,url,parentkey)	updateComplete
LabelField		remNode(index)	itemClick
editable			itemEditBeginning
allowMultipleSelection			itemEditBegin
dragEnabled			itemEditEnd
dragMoveEnabled			itemFocusIn
dropEnabled			itemFocusOut
nodes			

xpertGrid: IxpertDataControl

Las grillas manejan 2 juegos de datos coordinados, el primero identifica las columnas, y el segundo los registros de datos con los campos para cada columna.

Propiedades	Estilo	Métodos	Eventos
editable		clear()	creationComplete
allowMultipleSelection		clearColumns()	updateComplete
dragEnabled		clearRows()	itemClick
dragMoveEnabled		addColumn(key,label,editable)	itemEditBeginning
dropEnabled		addRow()	itemEditBegin
rows		getValue(row,col)	itemEditEnd
		setValue(row,col,value)	itemFocusIn
			itemFocusOut

xpertChartC: IxpertDataControl

Las graficas manejan 2 juegos de datos coordinados, el primero identifica las series, y el segundo los puntos a graficar dentro de cada serie.

Propiedades	Estilo	Métodos	Eventos
points		clear()	creationComplete
		clearSeries()	updateComplete
		clearPoints()	
		addSerie(key,label)	
		addPoint(label)	
		getValue(point,serie)	
		setValue(point,serie,value)	

xpertChartL: IxpertDataControl

Las graficas manejan 2 juegos de datos coordinados, el primero identifica las series, y el segundo los puntos a graficar dentro de cada serie.

Propiedades	Estilo	Métodos	Eventos
points		clear()	creationComplete
		clearSeries()	updateComplete
		clearPoints()	
		addSerie(key,label)	
		addPoint(label)	
		getValue(point,serie)	
		setValue(point,serie,value)	

xpertChartP: IxpertDataControl

Las graficas manejan 2 juegos de datos coordinados, el primero identifica las series, y el segundo los puntos a graficar dentro de cada serie.

Propiedades	Estilo	Métodos	Eventos
points		clear() clearSeries() clearPoints() addSerie(key,label) addPoint(label) getValue(point,serie) setValue(point,serie,value)	creationComplete updateComplete

xpertMenuBar: IxpertDataControl

Los datos de la barra de menú representan los ítems a mostrar en el mismo (estructura de árbol) y el vínculo al que apuntan

Propiedades	Estilo	Métodos	Eventos
nodes		clear() addNode(key,label,selected,url,parentkey) remNode(index)	creationComplete updateComplete itemClick

xpertShelf: IxpertDataControl

Los datos de la galería representan las imágenes a mostrar en la misma

Propiedades	Estilo	Métodos	Eventos
		clear() addImage(url) remImage(index)	creationComplete updateComplete itemClick

xpertMap: IxpertDataControl

Los datos del mapa determina los puntos (latitud, longitud, descripción) visibles en el mapa

xpertTextRtf: IxpertValueControl

El valor del editor de texto es su texto con el formato

xpertTextArea: IxpertValueControl

El valor del área de texto es su texto con el formato

xpertMask: IxpertValueControl

El valor del cuadro de texto con mascara es su texto sin la máscara.

xpertColor: IxpertValueControl

El valor representa el color seleccionado como un número hexadecimal

xpertSelector: IxpertDataControl

Los datos determinan los ítems seleccionados y los disponibles a seleccionar (clave y descripción)

xpertUploader: IxpertControl

xpertAccordionH: IxpertMultiContainer, IxpertDataControl

Los datos determinan las secciones y opcionalmente el link al formulario contenido del acordeón.

Propiedades	Estilo	Métodos	Eventos
selectedIndex		clear()	creationComplete
ValueField		addTab(key,label)	updateComplete
LabelField		remTab(index)	change
SourceField			

xpertAccordionV: IxpertMultiContainer, IxpertDataControl

Los datos determinan las secciones y opcionalmente el link al formulario contenido del acordeón.

Propiedades	Estilo	Métodos	Eventos
selectedIndex		clear()	creationComplete
ValueField		addTab(key,label)	updateComplete
LabelField		remTab(index)	change
SourceField			

xpertDividerH: IxpertMultiContainer, IxpertDataControl

Los datos determinan las secciones y opcionalmente el link al formulario contenido del divider.

Propiedades	Estilo	Métodos	Eventos
selectedIndex		clear()	creationComplete
ValueField		addTab(key,label)	updateComplete
LabelField		remTab(index)	
SourceField			

xpertDividerV: IxpertMultiContainer, IxpertDataControl

Los datos determinan las secciones y opcionalmente el link al formulario contenido del divider.

Propiedades	Estilo	Métodos	Eventos
selectedIndex		clear()	creationComplete
ValueField		addTab(key,label)	updateComplete
LabelField		remTab(index)	
SourceField			

xpertTab: IxpertMultiContainer, IxpertDataControl

Los datos determinan las solapas y opcionalmente el link al formulario contenido del tabfolder.

Propiedades	Estilo	Métodos	Eventos
selectedIndex ValueField LabelField SourceField		clear() addTab(key,label) remTab(index)	creationComplete updateComplete change

xpertBook: IxpertMultiContainer, IxpertDataControl

Los datos determinan las hojas y opcionalmente el link al formulario contenido del libro

Propiedades	Estilo	Métodos	Eventos
selectedIndex ValueField LabelField SourceField		clear() addTab(key,label) remTab(index)	creationComplete updateComplete change

xpertBox: IxpertContainer

Propiedades	Estilo	Métodos	Eventos
source		clear() load(url)	creationComplete updateComplete mouseOver mouseOut click

xpertPanel: IxpertContainer

Propiedades	Estilo	Métodos	Eventos
Source		clear() load(url)	creationComplete updateComplete mouseOver mouseOut click

xpertLine ← **xpertBox**
xpertTriangle ← **xpertBox**
xpertSquare ← **xpertBox**
xpertPentagon ← **xpertBox**
xpertHexagon ← **xpertBox**
xpertCircle ← **xpertBox**
xpertArrow ← **xpertBox**

Anexo Sintaxis de Eventos

La sintaxis que se utiliza para escribir los eventos es básicamente la de JavaScript o ActionScript (ECMA 262 en ambos casos), dispone de los siguientes elementos:

Estructuras de Control

break	Appears within a loop (for, for..in, for each..in, do..while, or while).
continue	Jumps past all remaining statements in the innermost loop and starts the next iteration of the loop as if control had passed through to the end of the loop normally.
do..while	Similar to a while loop, except that the statements are executed once before the initial evaluation of the condition.
else	Specifies the statements to run if the condition in the if statement returns false.
for	Evaluates the init (initialize) expression once and then starts a looping sequence.
for..in	Iterates over the dynamic properties of an object or elements in an array and executes statement for each property or element.
for each..in	Iterates over the items of a collection and executes statement for each item.
if	Evaluates a condition to determine the next statement to execute.
return	Causes execution to return immediately to the calling function.
throw	Generates, or *throws*, an error that can be handled, or *caught*, by a catch code block.
while	Evaluates a condition and if the condition evaluates to true, executes one or more statements before looping back to evaluate the condition again.

Operadores

Aritmetic		
+	addition	Adds numeric expressions.
--	decrement	Subtracts 1 from the operand.
/	division	Divides expression1 by expression2.
++	increment	Adds 1 to an expression.
%	modulo	Calculates the remainder of expression1 divided by expression2.
*	multiplication	Multiplies two numerical expressions.
-	subtraction	Used for negating or subtracting.
Arithmetic compound assignment		
+=	addition assignment	Assigns expression1 the value of expression1 + expression2.
/=	division assignment	Assigns expression1 the value of expression1 / expression2.
%=	modulo assignment	Assigns expression1 the value of expression1 % expression2.
*=	multiplication assignment	Assigns expression1 the value of expression1 * expression2.
-=	subtraction assignment	Assigns expression1 the value of expression1 - expression2.
Assignment		
=	assignment	Assigns the value of expression2 (the operand on the right) to the variable, array element, or property in expression1.
Bitwise		
&	bitwise AND	Converts expression1 and expression2 to 32-bit unsigned integers, and performs a Boolean AND operation on each bit of the integer parameters.
<<	bitwise left shift	Converts expression1 and shiftCount to 32-bit integers, and shifts all the bits in expression1 to the left by the number of places specified

		by the integer resulting from the conversion of shiftCount.
~	bitwise NOT	Converts expression to a 32-bit signed integer, and then applies a bitwise one's complement.
|	bitwise OR	Converts expression1 and expression2 to 32-bit unsigned integers, and places a 1 in each bit position where the corresponding bits of either expression1 or expression2 are 1.
>>	bitwise right shift	Converts expression and shiftCount to 32-bit integers, and shifts all the bits in expression to the right by the number of places specified by the integer that results from the conversion of shiftCount.
>>>	bitwise unsigned right shift	The same as the bitwise right shift (>>) operator except that it does not preserve the sign of the original expression because the bits on the left are always filled with 0.
^	bitwise XOR	Converts expression1 and expression2 to 32-bit unsigned integers, and places a 1 in each bit position where the corresponding bits in expression1 or expression2, but not both, are 1.
Bitwise compound assignment		
&=	bitwise AND assignment	Assigns expression1 the value of expression1 & expression2.
<<=	bitwise left shift and assignment	Performs a bitwise left shift (<<=) operation and stores the contents as a result in expression1.
|=	bitwise OR assignment	Assigns expression1 the value of expression1 | expression2.
>>=	bitwise right shift and assignment	Performs a bitwise right-shift operation and stores the result in expression.
>>>=	bitwise unsigned right shift and assignment	Performs an unsigned bitwise right-shift operation and stores the result in expression.
^=	bitwise XOR assignment	Assigns expression1 the value of expression1 ^ expression2.
Comment		
/*..*/	block comment delimiter	Delimits one or more lines of script comments.
//	line comment delimiter	Indicates the beginning of a script comment.
Comparison		
==	equality	Tests two expressions for equality.
>	greater than	Compares two expressions and determines whether expression1 is greater than expression2; if it is, the result is true.
>=	greater than or equal to	Compares two expressions and determines whether expression1 is greater than or equal to expression2 (true) or expression1 is less than expression2 (false).
!=	inequality	Tests for the exact opposite of the equality (==) operator.
<	less than	Compares two expressions and determines whether expression1 is less than expression2; if so, the result is true.
<=	less than or equal to	Compares two expressions and determines whether expression1 is less than or equal to expression2; if it is, the result is true.
===	strict equality	Tests two expressions for equality, but does not perform automatic data conversion.
!==	strict inequality	Tests for the exact opposite of the strict equality (===) operator.
Logical		
&&	logical AND	Returns expression1 if it is false or can be converted to false, and expression2 otherwise.
!	logical NOT	Inverts the Boolean value of a variable or expression.
||	logical OR	Returns expression1 if it is true or can be converted to true, and expression2 otherwise.
Other		
[]	array access	Initializes a new array or multidimensional array with the specified elements (a0, and so on), or accesses elements in an array.
as	as	Evaluates whether an expression specified by the first operand is a

		member of the data type specified by the second operand.
,	comma	Evaluates expression1, then expression2, and so on.
?:	conditional	Evaluates expression1, and if the value of expression1 is true, the result is the value of expression2; otherwise the result is the value of expression3.
delete	delete	Destroys the object property specified by reference; the result is true if the property does not exist after the operation completes, and false otherwise.
.	dot	Accesses class variables and methods, gets and sets object properties, and delimits imported packages or classes.
In	in	Evaluates whether a property is part of a specific object.
Instanceof	instanceof	Evaluates whether an expression's prototype chain includes the prototype object for function.
Is	is	Evaluates whether an object is compatible with a specific data type, class, or interface.
::	name qualifier	Identifies the namespace of a property, a method, an XML property, or an XML attribute.
new	new	Instantiates a class instance.
{}	object initializer	Creates a new object and initializes it with the specified name and value property pairs.
()	parentheses	Performs a grouping operation on one or more parameters, performs sequential evaluation of expressions, or surrounds one or more parameters and passes them as arguments to a function that precedes the parentheses.
/	RegExp delimiter	When used before and after characters, indicates that the characters have a literal value and are considered a regular expression (RegExp), not a variable, string, or other ActionScript element.
:	type	Used for assigning a data type; this operator specifies the variable type, function return type, or function parameter type.
typeof	typeof	Evaluates expression and returns a string specifying the expression's data type.
Void	void	Evaluates an expression and then discards its value, returning undefined.
String		
+	concatenation	Concatenates (combines) strings.
+=	concatenation assignment	Assigns expression1 the value of expression1 + expression2.
"	string delimiter	When used before and after characters, indicates that the characters have a literal value and are considered a string, not a variable, numerical value, or other ActionScript element.
XML		
@	attribute identifier	Identifies attributes of an XML or XMLList object.
{ }	braces (XML)	Evaluates an expression that is used in an XML or XMLList initializer.
[ ]	brackets (XML)	Accesses a property or attribute of an XML or XMLList object.
+	concatenation (XMLList)	Concatenates (combines) XML or XMLList values into an XMLList object.
+=	concatenation assignment (XMLList)	Assigns expression1, which is an XMLList object, the value of expression1 + expression2.
	delete (XML)	Deletes the XML elements or attributes specified by reference.
..	descendant accessor	Navigates to descendant elements of an XML or XMLList object, or (combined with the @ operator) finds matching attributes of descendants.
.	dot (XML)	Navigates to child elements of an XML or XMLList object, or (combined with the @ operator) returns attributes of an XML or XMLList object.
( )	parentheses (XML)	Evaluates an expression in an E4X XML construct.
< >	XML literal tag delimiter	Defines an XML tag in an XML literal.

Constantes Globales

Infinity : Number A special value representing positive Infinity.
-Infinity : Number A special value representing negative Infinity.
NaN : Number A special member of the Number data type that represents a value that is "not a number" (NaN).
undefined : * A special value that applies to untyped variables that have not been initialized or dynamic object properties that are not initialized.

Funciones Globales

Array(... args):Array Creates a new array.
Boolean(expression:Object):Boolean Converts the expression parameter to a Boolean value and returns the value.
decodeURI(uri:String):String Decodes an encoded URI into a string.
decodeURIComponent(uri:String):String Decodes an encoded URI component into a string.
encodeURI(uri:String):String Encodes a string into a valid URI (Uniform Resource Identifier).
encodeURIComponent(uri:String):String Encodes a string into a valid URI component.
escape(str:String):String Converts the parameter to a string and encodes it in a URL-encoded format, where most nonalphanumeric characters are replaced with % hexadecimal sequences.
int(value:Number):int Converts a given numeric value to an integer value.
isFinite(num:Number):Boolean Returns true if the value is a finite number, or false if the value is Infinity or -Infinity.
isNaN(num:Number):Boolean Returns true if the value is NaN(not a number).
isXMLName(str:String):Boolean Determines whether the specified string is a valid name for an XML element or attribute.
Number(expression:Object):Number Converts a given value to a Number value.
Object(value:Object):Object Every value in ActionScript 3.0 is an object, which means that calling Object() on a value returns that value.
parseFloat(str:String):Number Converts a string to a floating-point number.
parseInt(str:String, radix:uint = 0):Number Converts a string to an integer.
String(expression:Object):String Returns a string representation of the specified parameter.
trace(... arguments):void Displays expressions, or writes to log files, while debugging.
uint(value:Number):uint

Converts a given numeric value to an unsigned integer value.
unescape(str:String):String Evaluates the parameter str as a string, decodes the string from URL-encoded format (converting all hexadecimal sequences to ASCII characters), and returns the string.
XML(expression:Object):XML Converts an object to an XML object.
XMLList(expression:Object):XMLList Converts an object to an XMLList object.

Anexo Instalación y Configuración

Introducción:

RIATools es una herramienta para el desarrollo de la capa de presentación de portales y aplicaciones web o aplicaciones desktop integrable a prácticamente cualquier tipo de aplicación de backend (donde Ud. implementa las reglas de negocio y acceso a datos).

Este anexo presenta los pasos y consideraciones a seguir a los efectos de instalar dichas herramientas en un determinado entorno.

Prerrequisitos:

Edición WEB

Para instalar la edición web de las RIATools es requerido disponer de un servidor WEB además de los archivos incluidos en la distribución. Adicionalmente se requiere la infraestructura que utilice su aplicación de backend (habitualmente un servidor de aplicaciones o contenedor de servelets y un servidor de base de datos).

Los siguientes servidores web y de aplicaciones son soportados:

- MS IIS (http://www.iis.net/default.aspx?tabid=1)
- Cassini (http://www.asp.net/downloads/archived/cassini/)
- Apache (http://httpd.apache.org/)
- Tomcat (http://tomcat.apache.org/)
- OAS (http://www.oracle.com/technology/products/ias/index.html)
- JBoss (http://labs.jboss.com/projects/download/)
- JRun (http://www.adobe.com/products/jrun/)

Dependiendo de qué versión de loader y bridge desea utilizar puede ser necesario algún modulo adicional (por ejemplo PHP para IIS o Apache). Puede optar entre las siguientes versiones de loader y bridges:

ASPX – Para .NET o Gx .NET sobre MS IIS y Cassini

JSP – Para J2EE o Gx JAVA sobre Tomcat, OAS, JBoss o JRun

PHP – Sobre MSIIS o Apache + el modulo PHP correspondiente (solo para backends LAMP/WAMP)

Edición Desktop

Para instalar la edición desktop de las RIATools solo se requieren los archivos incluidos en la distribución de las mismas. Ud., puede optar por las siguientes alternativas desktop: .NET, JAVA o AIR.

Estructura de Directorios:

La estructura más común de directorios es la siguiente:

/root
/root/astifacts
/root/assets

Para ver que archivo debe incluir en cada directorio consulte la sección “Componentes de las RIA Tools”

Componentes de las RIATools:

Los componentes incluidos en la distribución de las RIATools son los siguientes:

Ambiente	Archivo	Descripción
Todos	designer.swf	Binario Flash del diseñador WYSIWYG de frontend RIAs
Todos	designerlt.swf	Binario Flash del diseñador LITE de frontend RIAs
Todos	styler.swf	Binario Flash del editor de temas y estilos de frontend RIAs
Todos	runner.swf	Binario Flash del ejecutor de frontend RIAs
Todos	config.xml	Archivo de configuración de RIATools
Todos	styles.xml	Archivo de temas y estilos dinámicos de RIATools
Todos	artifacts/sample.xml	Ejemplo de frontend RIA construido con el designer
Todos	assets/image.png	Asset del designer para xpertImages
Todos	assets/clip.swf	Asset del deisgner para xpertClips
Todos	assets/add.png	Asset usado en ejemplo: simple.xml
Todos	assets/delete.png	Asset usado en ejemplo: simple.xml
Todos	assets/edit.png	Asset usado en ejemplo: simple.xml
Todos	assets/refresh.png	Asset usado en ejemplo: simple.xml
Todos	assets/back.png	Asset usado en ejemplo: simple.xml
Todos	assets/forward.png	Asset usado en ejemplo: simple.xml
WEB	designer.html	Wrapper HTML de designer.swf
WEB	designerlt.html	Wrapper HTML de designerlt.swf
WEB	styler.html	Wrapper HTML de styler.swf
WEB	runner.html	Wrapper HTML de runner.swf
WEB .NET	loader.aspx	Loader .NET
WEB .NET	uploader.aspx	Uploader .NET
WEB .NET	bridge.aspx	Bridge .NET
WEB J2EE	loader.jsp	Loader JSP
WEB J2EE	uploader.jsp	Uploader JSP
WEB J2EE	bridge.jsp	Bridge JSP
WEB PHP	loader.php	Loader PHP
WEB PHP	uploader.php	Uploader PHP
WEB PHP	bridge.php	Bridge PHP
WEB PHP	xpertIO.php	Rutinas para manejo de archives, strems, y request/response
WEB PHP	xpertXML.php	Rutinas para manipulación de XML
WEB PHP	xpertData.php	Rutinas para acceso a datos en repositorios MySQL
Desktop WIN	designer.exe	Wrapper HTML de designer.swf
Desktop WIN	designerlt.exe	Wrapper HTML de designerlt.swf
Desktop WIN	styler.exe	Wrapper HTML de styler.swf
Desktop WIN	runner.exe	Wrapper HTML de runner.swf
Desktop AIR	designer.air	Wrapper AIR de designer.swf
Desktop AIR	designerlt.air	Wrapper AIR de designerlt.swf
Desktop AIR	styler.air	Wrapper AIR de styler.swf
Desktop AIR	runner.air	Wrapper AIR de runner.swf
Desktop JAVA	designer.class	Wrapper JAVA de designer.swf
Desktop	designerlt.class	Wrapper JAVA de designerlt.swf

JAVA		
Desktop JAVA	styler.class	Wrapper JAVA de styler.swf
Desktop JAVA	runner.class	Wrapper JAVA de runner.swf

Configuración de RIATools:

La configuración de las RIATools se realiza editando el archivo “config.xml”, el contenido de este archivo suele ser algo similar a esto:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<config ACTION="CONFIG" STATUS="OK" STARTUP="artifacts/sample.xml">
        <licence key1="" key2="" type="EVALUATION" />
        <uploader url="uploader.aspx" root="assets" />
        <loader url="loader.aspx" root="artifacts" />
        <bridge url="bridge.aspx" type="GXV9"/>
</config>
```

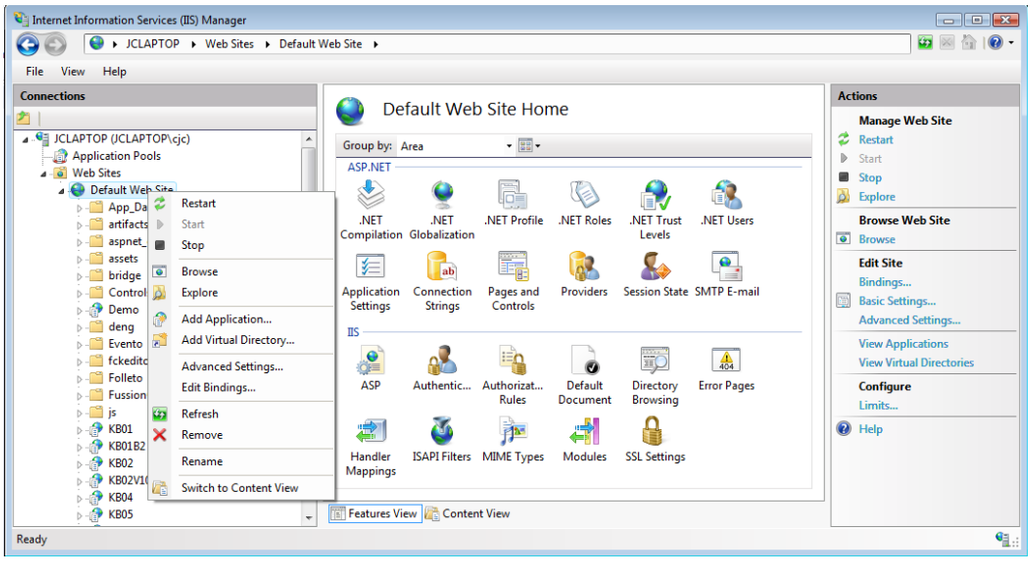
Este archivo debe estar ubicado en el mismo directorio que los binarios swf.

Para configurar las RIATools en un servidor web, lo ideal es crear una nueva aplicación en el servidor web y copiar los archivos mencionados en la sección 4 en la estructura de directorios indicada en la sección 3. Si esta instalando las RIA Tools en IIS consulte la sección 6, si esta instalándolas en Tomcat o similares, consulte la sección 7.

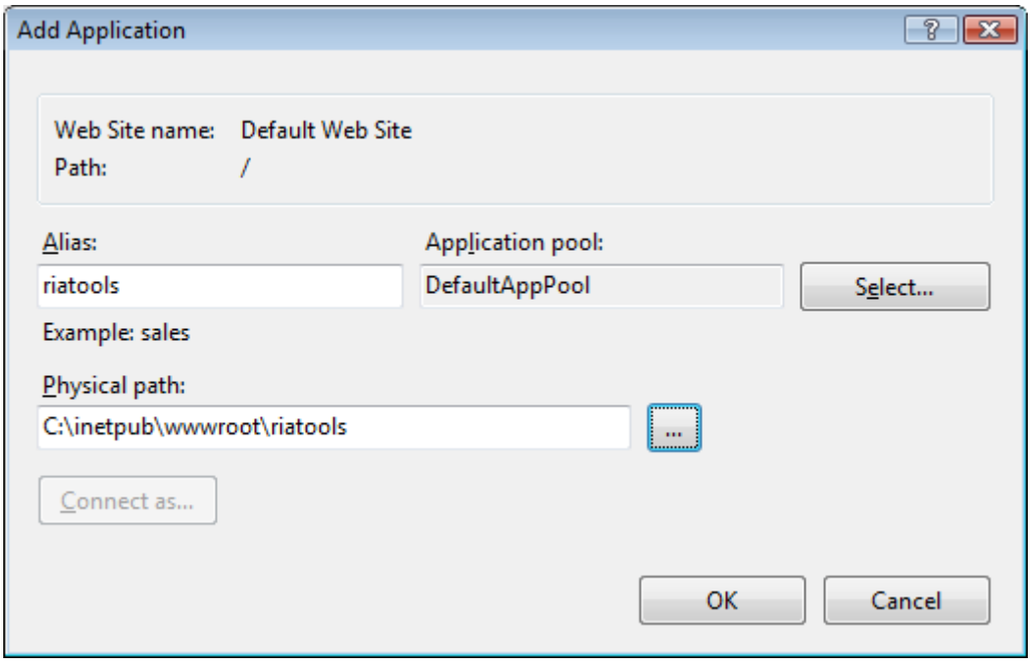
NOTA: Dado que el designer salva los frontend RIA en el directorio indicado por la propiedad “root” del nodo “loader” es requerido disponer de permiso de escritura sobre esa carpeta, por defecto la carpeta “artifacts”

Configuración de RIATools sobre IIS:

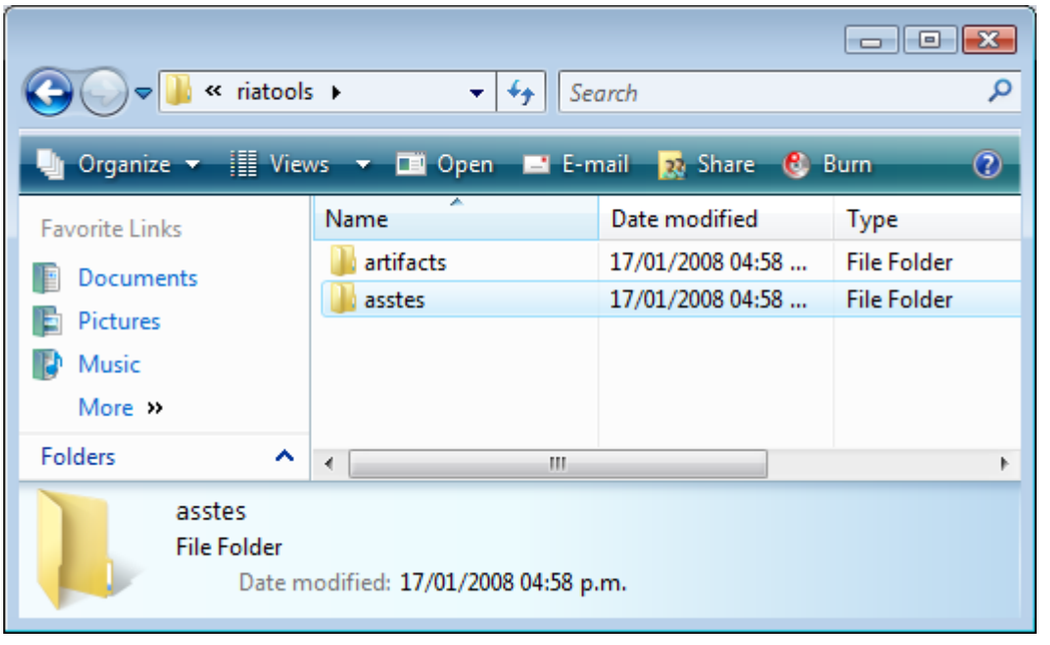
En la consola de administración de IIS, debajo del servidor web que vaya utilizar (normalmente descripto como “Default Web Site”) seleccione la opción “Add Application” desde el menú contextual.



Indique el alias de la aplicación y la ruta de directorio que servirá como raíz de la aplicación.



Cree las carpetas “artifacts” y “assets” debajo de dicho directorio:



Copie los archivos incluidos en la distribución indicados en la sección 4 respetando las carpetas de cada uno.

Si utiliza otros nombres de carpetas para los artefactos RIA y las imágenes, deberá cambiar las propiedades root del nodo loader y uploader en el config.xml. Si usa los nombres sugeridos no es necesario cambiar la configuración por defecto.

Si experimenta problemas ubicando los componentes del middleware (laoder, uploader y bridge) pruebe usando las rutas absolutas en el config.xml

Por ejemplo, si tiene IIS instalado en localhost (local en un equipo)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<config ACTION="CONFIG" STATUS="OK" STARTUP="artifacts/sample.xml">
        <licence key1="" key2="" type="EVALUATION" />
        <uploader url=" http://localhost/riatools/uploader.aspx" root="
http://localhost/riatools/assets" />
        <loader url=" http://localhost/riatools/loader.aspx" root="
http://localhost/riatools/artifacts" />
        <bridge url="http://localhost/riatools/bridge.aspx" type="GXV9"/>
</config>
```

Para probar si su instalación quedo correcta ingrese las siguientes URL (cambie la ruta acorde a la ubicación de su instalación)

http://localhost/riatools/designer.html
http://localhost/riatools/designerlt.html
http://localhost/riatools/styler.html
http://localhost/riatools/runner.html
http://localhost/riatools/loader.aspx
http://localhost/riatools/bridge.aspx

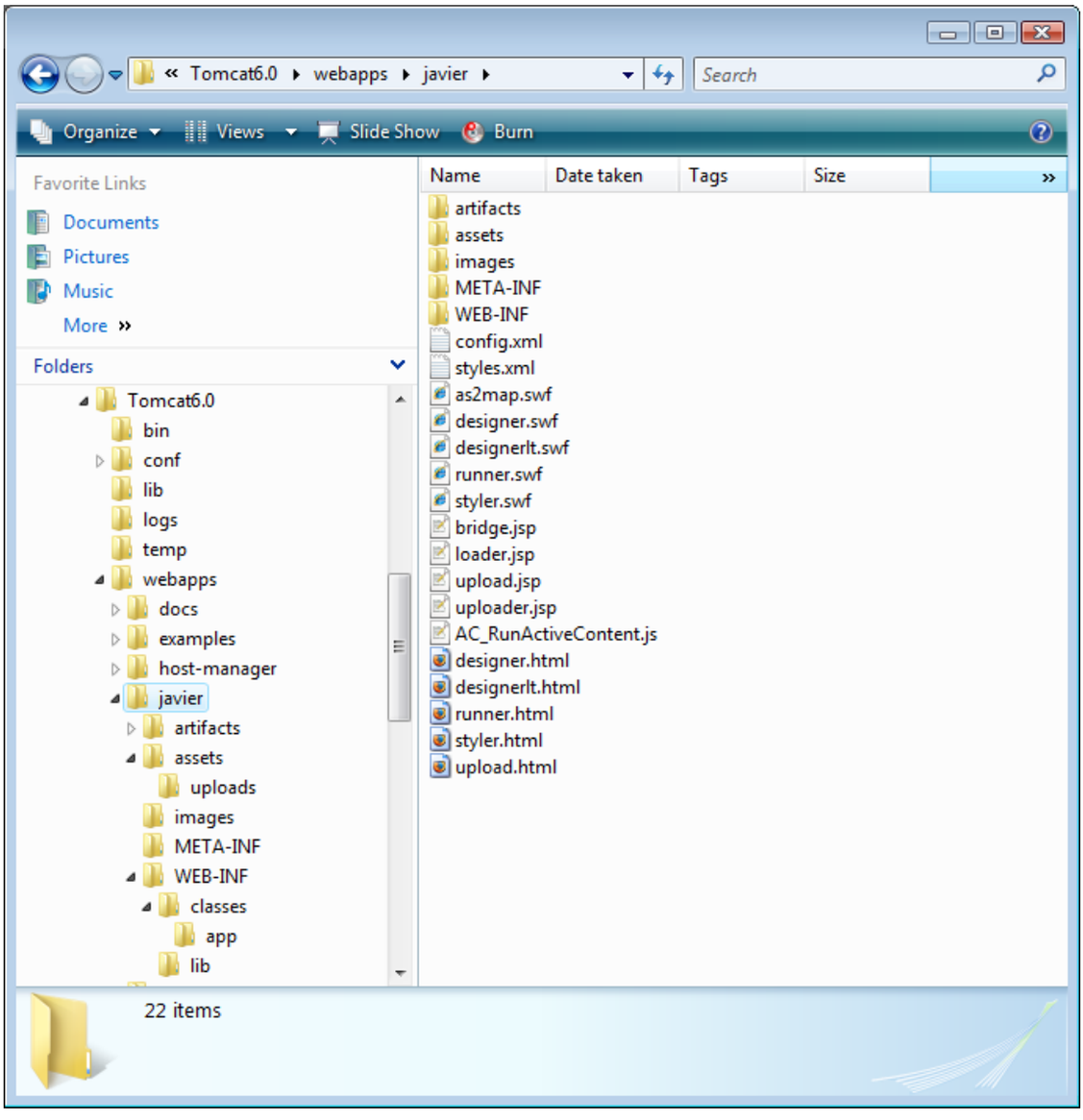
Configuración de RIATools sobre Tomcat:

Para configurar las RIATools sobre Tomcat, debe obtener e instalar primero las siguientes API JAVA distribuidas en forma libre (no hemos incluido los mismos en la distribución para reducir su tamaño y darle la libertad de usar la versión que prefiera de los mismos):

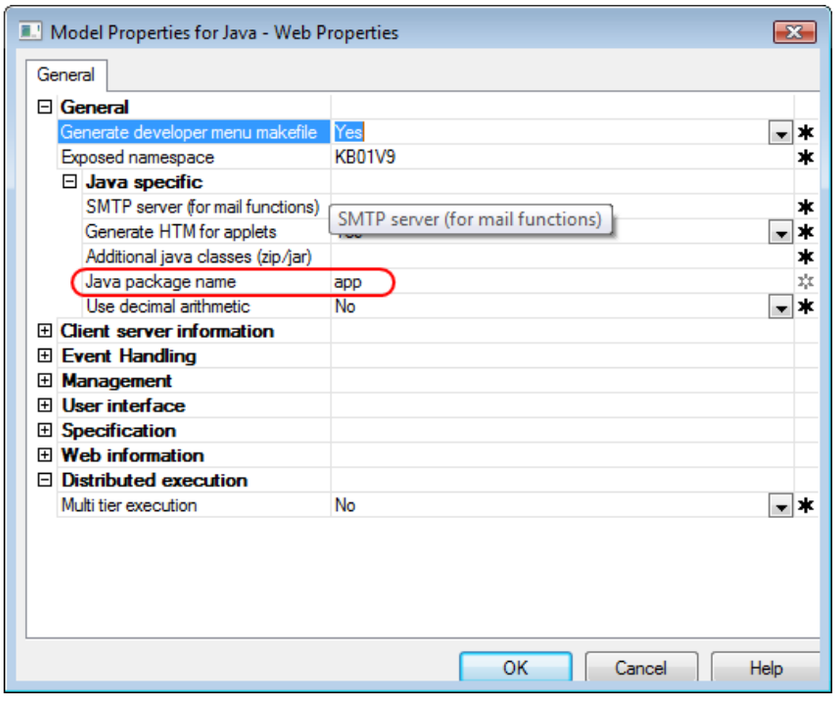
jdom.jar (para el manejo de XML, descargable desde aquí)
commons-io.jar (para el manejo de archives, descargable desde aquí)
commons-fileupload.jar (para el upload de archivos, descargable desde aquí)

Estos archivos deben ubicarse en la carpeta commons/lib (o /lib, varía según la versión de Tomcat que utilice) de la instalación de Tomcat.

Por último, coloque los archivos de la distribución de RIATool en el directorio raíz de la webapp donde haya hecho el deployment de su aplicación de backend. Debería obtener una estructura de directorios y archivos similar a la presentada en la siguiente imagen:



IMPORTANTE: Si va a utilizar como backend de su aplicación RIATool una aplicación construida con Genexus v8 o v9, es necesario que le asigne "**app**" a la propiedad "Java package name" del modelo JAVA (como se puede observar en la imagen). Genexus -por defecto- genera las clases que implementan los objetos de su aplicación en el default package, y por una restricción de seguridad de la JVM, no esta permitido instancia dinámicamente por reflection clases del default package, dado que el bridge.jsp utiliza esta técnica para ejecutar los procs GX, es requerido hacer esta modificación inocua en su modelo Genexus para que genere las clases en el package "app" donde el bridge.jsp de las RIATools espera encontrar dichas clases a instanciar. Adicionalmente, es necesario abrir el API GXClassR.JAR, y reemplazar la ruta /org/apache/commons/fileupload por la existente dentro del API commons-fileupload.jar debido a que Genexus ya utiliza esta API pero la misma no se encuentra completa.



Troubleshooting:

To do